Издательство «Знание»

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается ежемесячно с 1988 г.

Эти разные Бейсики

1992

4

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

4/1992

Издается
ежемесячно
с 1988 г.

ЭТИ РАЗНЫЕ БЕЙСИКИ

В номере:

В.Ф.Очков
ЯЗЫК ПРОГРАММИРОВАНИЯ QBASIC

А.В.Петроченков
РАЗРЕШИТЕ ПРЕДСТАВИТЬСЯ: DOS 5.0!

М.В.Михулин
ПРОГРАММА DEC

РУБРИКИ

БК за рога
Обмен опытом

Издательство
«Знание»
Москва
1992

ББК 32.97
Э 90

Авторы выпуска

ОЧКОВ ВАЛЕРИЙ ФЕДОРОВИЧ — кандидат технических наук, доцент Московского энергетического института.

ПЕТРОЧЕНКОВ АЛЕКСАНДР ВАСИЛЬЕВИЧ — технический писатель и журналист, автор статей о программных продуктах для персональных компьютеров.

МИХУЛИН МИХАИЛ ВЛАДИМИРОВИЧ — инженер-механик, программист.

НАДЕЖИН АЛЕКСЕЙ МИХАЙЛОВИЧ — инженер, программист.

НИЧИПУРЕНКО Э. В., НОВИКОВ Ф.В., СУХОВ Д.С., ТАРАСОВ В.Б.— наши подписчики.

РЕДАКТОР Б.М.ВАСИЛЬЕВ

ISBN 5-07-002497-3

© Издательство «Знание», 1992 г.

«Вы думаете, я знаю Бейсик! Нет, я не имел счастия воспользоваться таким воспитанием. Мой отец был мерзавец, скотина. Он и не думал меня выучить Бейсику. Я был тогда еще ребенком, меня легко было приучить — стоило только посечь хорошенько, и я бы знал, я бы непременно знал.»

Почти Н.Гоголь

В.Ф.Очков

ЯЗЫК ПРОГРАММИРОВАНИЯ QBASIC

УВЕРТЮРА (Избранные места из каталога Microsoft)

Новая версия MS-DOS — наиболее значительная из существующих. Это мощная, высокостабильная операционная система для работы на персональных компьютерах самых различных конфигураций. MS-DOS 5.0 обеспечивает схему управления памятью, которая значительно сокращает объем ОЗУ, занимаемый самой DOS, и освобождает часть памяти до 640 Кбайт в младших адресах. Новая версия также поддерживает разделы жесткого диска размером более 32 Мбайт и включает технологию защиты по питанию для переносных компьютеров.

Новая версия MS-DOS проще в использовании, чем ее предшественницы. Она включает новую утилиту для автономной инсталляции и для инсталляции в сети.

Встроенная система подсказки дает возможность пользователям получать справочную информацию по командам MS-DOS простым нажатием специальной клавиши. Кроме того, MS-DOS 5.0 содержит новую, улучшенную оболочку пользователя, совместимую с интерфейсом Windows 3.0, благодаря чему пользователь мгновенно знакомится со всеми преимуществами графического интерфейса пользователя. Помимо оболочки, в систему входит переключатель задач, позволяющий пользователям быстро переходить от одной прикладной задачи к другой, не дожидаясь выхода из одной программы и загрузки другой.

Новую версию MS-DOS усиливает ряд новых утилит. Мощные утилиты для восстановления удаленных файлов, быстрого форматирования, редактирования строки и повторного вызова команд значительно упрощают рутинные задачи.

Имеется также полноэкранный текстовый редактор, который освобождает от необходимости использования EDLIN для редактирования ASCII-файлов. Редактор использует раскрывающееся меню, может работать с мышью и содержит встроенную документацию с гипертекстовыми связями.

И последний аккорд увертюры. Вместо прежнего GW-BASIC-интерпретатора новая версия MS-DOS включает среду программирования QBASIC — тему нашей статьи.

СРЕДА ПРОГРАММИРОВАНИЯ QBASIC В РУСИФИЦИРОВАННОЙ ВЕРСИИ MS-DOS 5.0

Есть у меня шестерка слуг
Проворных, удалых.
И все, что вижу я вокруг,
Все узнаю от них.
Они по зову моему
Являются ко мне.
Their names are What,
and Why, and When, and
How, and Where, and Who.

Р.Киплинг

С первым эпиграфом все ясно из названия статьи. Второй эпиграф — это фрагмент стихотворения из сказки о любопытном слоненке, которое автор когда-то учил на школьных уроках английского, запоминая слова «Что — What», «Почему — Why», «Когда — When», «Как — How», «Где — Where» и «Кто — Who», и от которого в памяти осталось лишь переводное маршаковское начало и оригинальный киплинговский конец. В этом эпиграфе стоит обратить внимание и на форму, и на содержание.

Сначала о форме

Текст статьи, как и эпиграф, также двуязычен — настолько часто в нем встречаются английские термины среди русских фраз. Автор был бы рад, идя навстречу просьбам некоторых читателей, не знающих английского (если таковые имеются среди читателей этого журнала), совсем отказаться от иностранных слов:

Но *принтер, плотер, файл, дискета,*
Всех этих слов на русском *нету!*
А вижу я, винюсь пред вами,
Что уж и так мой бедный слог
Пестреть гораздо б меньше мог
Иноплеменными словами,
Хоть и заглядывал я встарь
В Академический словарь.

Так, наверно, ответил бы А.С.Пушкин славянофилу А.С.Шишкову в наше компьютерное время. Это немцы или французы имеют моральное право сетовать на то, что международным языком информатики стал английский. Когда-то у нас кибернетику обозвали публичной девкой империализма. Она, конечно, обиделась и по-русски долго не говорила. Фирма Microsoft разработала руси-

фицированную версию MS-DOS 5.0. QBASIC в составе этой версии DOS.

Теперь о содержании эпиграфа. Читатель, по-видимому, немало наслышан о языках программирования. Но QBASIC это не язык, а среда программирования. Это означает, что программист, сидя за дисплеем, не остается наедине с бесчисленными операторами, командами и служебными словами языка. Не утонуть в море сведений о правилах программирования ему теперь помогает «шестерка слуг проворных, удалых». Вот их русские имена в среде QBASIC: **Файл, Редактирование, Просмотр, Поиск, Запуск, Отладка, Параметры и Справка** (рис. 1). Здесь даже не шестерка, а восьмерка, но это дела не меняет.

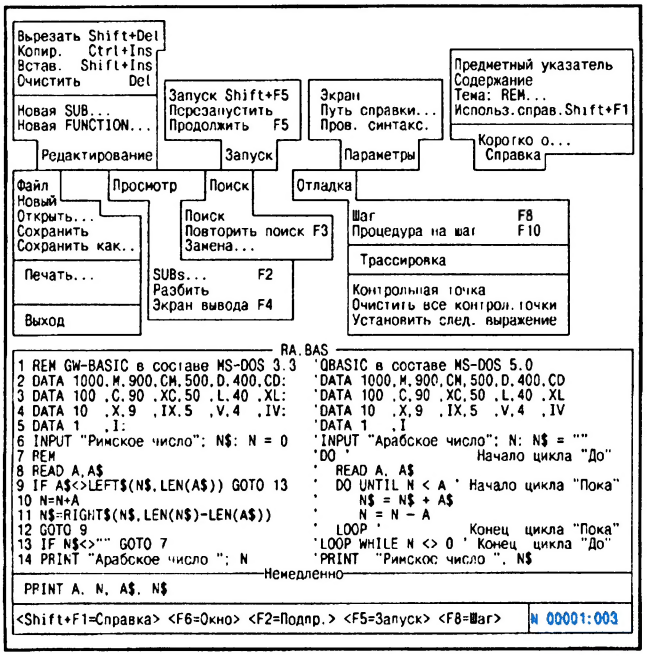


Рис. 1. Опытный программист сразу увидит в этой картинке почерк фирмы Microsoft. Некоторых, правда, удивят слова Файл, Просмотр, Поиск вместо более привычных File, View, Search. Других порадуют — нет необходимости вместе с информатикой учить и английский язык

Включив IBM PC с дисковой операционной системой MS-DOS 5.0 и запустив файл *qbasic.exe* в каталоге DOS, можно, как выражаются пользователи ЭВМ, войти в нужную программную среду, отличительным признаком которой будет некая цепочка слов, появляющаяся в верхней (или нижней) части дисплея.

С подобными словами-слугами читатель, конечно, уже встречался, знакомясь, например, с надстройкой над DOS Norton Commander (там «прислуживают» Help, User, View, Edit, Copy, RenMov, Mkdir, Delete, Menu, Quit), а также с текстовым редактором Лексикон («Текст», «Абзац», «Фрагмент», «Страницы», «Найти», «Заменить», «Отметить», «Перейти», «Шрифт», «ДОС», «Выход»). Или вспомним старого приятеля — язык программирования GW-BASIC, входящий в состав MS-DOS версии 3.30. С ним помогают работать клавиши F1 — LIST, F2 — RUN, F3 — LOAD, F4 — SAVE, F5 — CONT, F6 — LPT1, F7 — TRON, F8 — TROFF, F9 — KEY и F10 — SCREEN. Названия функциональных клавиш выстраиваются в нижней части дисплея после запуска программы gwbasic.exe в пакете MS-DOS 3.30.

Расшифровка значений этих слов — «блюд» — главного меню соответствующих программных сред — это наполовину освоенный программный продукт.

Читатель знает, чтобы заставить слово-слугу «проворно и удало» работать, нужно навести на него курсор и поработать клавишами «Стрелка влево» и «Стрелка вправо», а затем нажать клавишу Enter. В среде QBASIC, как и в других популярных программных средах фирмы Microsoft, текстовый процессор MS Word, интегрированный пакет MS Works, среда программирования QuickBASIC (QBASIC отличается от нее примерно так, как буква Q отличается от слова Quick), меню активизируется нажатием клавиши Alt.

После этого также можно поработать клавишами «Стрелка вправо» и «Стрелка влево», а затем нажать клавишу Enter. Из «окурсоренного» слова выпадет колонка с именами новых «слуг», объединенных во вспомогательном меню. Слова этого нового меню, в свою очередь, могут быть заголовками новых меню и т.д. Многоточие за названием «блюда» свидетельствует о том, что это не команда, а название нового меню. Меню почти всех программных сред построено в виде дерева, карабкаясь по ветвям которого пользователь ЭВМ может быстро найти то, что ему нужно.

Когда к щиту управления какой-нибудь промышленной установкой подходит очередной сменный оператор, то он, как правило, первым делом проверяет исправность всех лампочек, табло, звонков, сирен, на сигналы которых он должен будет потом реагировать. Для этого на щите нажимается специальная кнопка — и все табло горит, а звонки звенят. Неисправность же сразу выявляется.

Так и мы мысленно нажмем некую гипотетическую клавишу, чтобы из названий главного меню среды QBASIC «вывалились» вспомогательные меню (см. рис. 1) и стали видны все «слуги и прислужники». Для экономии места мы «вывалили» из главного меню все таблички вспомогательных меню не только вниз от полоски названий «блюд» главного меню, но и вверх. Рассказ о них, как и было обещано, составит сердцевину нашего повествования.

Файл. Группа команд, собранная здесь, — это для среды QBASIC окно в окружающий мир. Эти команды осуществляют связь машины с ее периферией — с дисководами для жестких и гибких дисков, на которых записываются программы, с при-

нтером для распечатки текстов программ, с дисковой операционной системой. Режим «Новый» очищает ОЗУ машины от старой программы. На языке GW-BASIC эту работу выполняет оператор NEW. Новая программа после этого может быть введена в компьютер двумя путями: набором текста программы на клавиатуре или вызовом ее с дисков. Чтобы считать программу с диска, нужно открыть программы ("Открыть...") и убедиться, что в появившемся списке есть ее имя. Если его нет, то сменить диск или каталог на нем. На языке GW-BASIC эту работу выполняет оператор FILES. Но вот имя нужной программы найдено. На него наводится курсор, нажимается клавиша Enter, и программа появляется на экране дисплея. На языке GW-BASIC для этой работы предусмотрен оператор LOAD, за которым в кавычках нужно написать имя вызываемой программы. Режим «Сохранить» (оператор SAVE в языке GW-BASIC) необходим в том случае, ежели требуется модернизированную старую программу вернуть на диск. Если имя программы и (или) место ее хранения в архивной памяти меняются, то понадобится команда «Сохранить как...». Машина, помня о забывчивости человека, зорко следит за тем, чтобы в архивной памяти оставалась копия программы, хранившейся в ОЗУ. И если человек захочет удалить ее из оперативной памяти командой «Новый» или «Выход» (см. далее), то услужливая машина напомнит ему о необратимости такой операции для старой программы и предложит записать ее на диск. Впрочем, человек от этого может отказаться, посчитав, что старая программа ему больше не понадобится или экономя место на дисках. Режим «Печать...» распечатывает на принтере текст программы: всю программу, подпрограммы, функции или только то, что выделено на экране дисплея. На языке GW-BASIC эта работа выполняется оператором LLIST. Наведя курсорную рамку на «Выход» и нажав Enter, мы покинем гостеприимную среду QBASIC и вернемся в MS-DOS 5.0 или в надстройку над DOS (Norton Commander или что-то другое), откуда среда QBASIC была вызвана.

Редактирование. Посмотрите на стол некомпьютеризированного редактора и вы, конечно, увидите там ножницы и клей. В арсенале среды QBASIC ножницы ("Вырезать") и клей ("Вставить") дополнены мечтой многих редакторов — копировальной машинкой ("Копировать"). Чтобы из текста программы вырезать для копирования или переноса какой-то участок, то его нужно сначала выделить цветом, нажимая клавиши перемещения курсора и притопив предварительно клавишу Shift. Курсор при этом будет «ползти» в заданном направлении, меняя цвет фона отдельных букв (если нажимать клавиши «Стрелка влево» или «Стрелка вправо») или целых строк ("Стрелка вниз" или «Стрелка вверх»). Выделенный таким образом текст можно из программы «Вырезать», а затем «Вставить» в любое место программы, куда предварительно переведен курсор. По выделенному фрагменту текста можно сделать штемпель ("Копировать"), орудовать которым ("Вставить") можно сколько душе угодно, предварительно передвинув в нужное место курсор. Редактор среды QBASIC, конечно, не такой мощный, как Лексикон, MS Word, Фотон или Chi-Writer, но вполне достаточный для работы с текстами. Автор писал эту статью в среде QBASIC, отключив предварительно контроль за синтаксисом строк (см. далее). Иначе машина бы оказывала «медвежьи услуги» — сообщала об ошибке (несоответствие правилам QBASIC'а) после каждого перевода строки. В этой колонке меню записаны две команды, не имеющие аналогий в языке GW-BASIC, — «Новая SUB...» и «Новая FUNCTION...», о которых мы поговорим чуть позже.

Просмотр. QBASIC, что существенно отличает его от предшественника GW-BASIC, вооружен таким мощным средством построения программ, как разбивка их на процедуры (SUBroutine) и функции (FUNCTION) с именами и локальными переменными. Об этой ценной особенности среды QBASIC речь впереди. Сейчас отметим, что выполнение позиции меню «SUBs...» выводит на экран дисплея названия всех задействованных в программе процедур и функций, а также и самой программы. Если требуется отредактировать или просто просмотреть какую-то процедуру или функцию, то достаточно в списке на ее имя навести курсор и нажать на клавишу Enter. Все. Нужный участок программы перед глазами программиста. Как понимает читатель, подобное поручение программист дает машине очень часто. Поэтому-то такой важной функцией нагружена вторая функциональная клавиша клавиатуры F2. Команда «Разбить» разбивает экран с текстом программы на две створки. Это очень удобно, так как позволяет редактировать программу, имея перед глазами какую-то отдаленную ее часть. Перебрасывается курсор из одной створки в другую аккордом Shift-F6. Повторное выполнение команды «Разбить» убирает вторую створку на экране дисплея. Команда «Экран вывода» позволяет выйти из окна редактирования программ в окно с результатами работы программы и посмотреть, что программа «натворила». Повторное выполнение этой команды возвращает программиста к окну редактирования.

Поиск. Здесь хранятся два очень нужных инструмента: «Поиск...» и «Замена...», позволяющие в программах вести поиск и при необходимости заменять определенную цепочку символов на новую. Командой «Замена...», кстати, можно «вылечивать» BASIC-программы, написанные на ранних версиях этого языка, от «детской болезни» — от коротких (в два байта) и поэтому «немых» имен переменных: A, A0, T и т.д. На языке QBASIC идентификаторы могут быть длиной до 40 символов. Дал команду «Замени цепочку »T" на цепочку «Temperatura» — и машина за доли секунды сделает программу более понятной. Делая замену, машина может переспрашивать человека: «В этом месте менять или пропустить?» Иначе компьютер заменит, например, букву T в служебных словах INPUT, PRINT на цепочку TEMPERATURA так, что программа потом ни на что не будет годна.

Запуск. Дело вроде бы простое, но и тут возможны нюансы, отмеченные в данном вспомогательном меню. Программу можно запустить с самого начала, «ткнув» в позицию «Запуск» (или взяв аккорд Shift-F5), продолжить ее выполнение («Продолжить» или F5) после прерывания аккордом Ctrl-Break или после выполнения оператора STOP. Команда «Перезапустить» позволяет выполнять программу пошагово.

Кроме того, программу можно вызвать на дисплей в среде QBASIC и запустить командной строкой DOS:

QBASIC [/B] [/EDITOR] [/G] [/H] [/MBF] [/NOHI] [/RUN] bc[исходный файл],

где расширения позволяют:

/B — использовать монохромный монитор с цветной графической картой; если у пользователя

цветной монитор, то параметр /B отобразит QBASIC в монохромном изображении;

/EDITOR — активизировать текстовой редактор MS-DOS 5.0 Editor (можно использовать сокращение: /ED);

/G — установить среду QBASIC для быстрого изменения режимов графика экрана CGA (действительно только для машин, использующих мониторы CGA);

/H — использовать максимально возможное число строк длявидеоадаптера;

/MBF — изменить функции QBASIC (CVS, CVD, MKS$, MKD$) для поддержки чисел в формате IEEE;

/NOHI — использовать монитор, который не поддерживаетповышенную яркость;

исходный файл — указать файл для вывода на дисплей.

/RUN — загрузить и запустить исходный файл QBASIC, прежде чем отобразить его на экране.

Отладка. Это один из самых ответственных моментов при создании новых программ и модернизации старых. Считается, что искусство программирования состоит не столько в умении составлять новые программы, сколько в способности программиста отыскивать в них ошибки. Сам английский термин a debug — «отладка» заслуживает того, чтобы о нем рассказать подробнее. A bug — по-английски «жучок», приставка de означает «удаление», так что сам термин Debug дословно можно перевести как «обезжучивание». Эта калька с английского не такая уж бессмысленная. Один из первых компьютеров стал барахлить из-за того, что в его электронных внутренностях запутался таракан, закоротил контакты, обуглился и как некий связист-камикадзе замкнул их, внеся тем самым изменения в электронную схему машины. Отлаживаемые программы можно не только «обезжучивать», но и «нажучивать». В стародавние времена, когда программы в ЭВМ вводили не их авторы, а девочки-перфораторщицы, и готовые тексты из-за этого содержали много «глупых» ошибок типа 5 вместо S, 1 вместо I и т.п., практиковался довольно оригинальный способ отладки по принципу «клин клином вышибают». Если нужно было узнать, сколько в программе «жучков», то в нее... вводилась еще сотня ошибок. Затем другой человек эти ошибки вылавливал. И если оказывалось, что он, к примеру, нашел 50 ошибок введенных специально и 10 первоначальных, то с определенной долей вероятности можно считать, что в программе осталось еще с десяток первородных «жучков». Кстати, подбор искусственных ошибок — это и наука, и искусство высокого класса, а сам процесс отладки очень трудоемкий. Недаром по-английски «to go bugs» означает «сойти с ума». В программе можно указать операторы, на которых машина будет прерывать выполнение программы ("Контрольная точка") и продолжать трансляцию после нажатия F5, F8 (пошаговое выполнение программы с «заходом» в процедуры и функции) или F10 (пошаговое выполнение программы без «захода» в процедуры и функции). Все эти «путы» контрольных точек, естественно, нужно будет убрать после отладки программы (см. команду «Очистить все контрольные точки»). «Трассировка» — это не команда, а переключатель. Если «ткнуть» в это слово и поставить тем самым слева от него точку, то интерпретатор QBASIC будет «прокручивать» программу, отмечая цветом букв и фона выполненные операторы, что позволяет быстро выявлять «хроническую болезнь» программ — зацикливание. Команда «Установить следующее выражение» позволяет при пошаговой прокрутке программ «перепрыгнуть» некоторые операторы, продолжив выполнение программы с отмеченного курсором места.

Параметры. Тут хранятся три инструмента настройки среды QBASIC. «Экран» — здесь QBASIC — проявляет высшую степень дружественности к человеку, позволяя ему оптимальным образом раскрасить экран дисплея: — выбрать приятный фон, неутомительный цвет букв и т.д. Режим «Путь справки...» дает компьютеру своеобразную «нить Ариадны» — указывает дорожки (D:\BASIC\QBASIC\HLP, например,) пробегая по которой, машина найдет файл qbasic.hlp со справочной информацией. О команде-переключателе «Проверка синтаксиса» было упомянуто ранее, когда говорилось о редакторских возможностях среды QBASIC. Если в меню слева от этих слов будет стоять точка, то среда QBASIC будет зорко следить за правильностью написания программ. Если при этом программист напишет INPUT и переведет курсор на новую строку, то машина откажется работать дальше, напомнив программисту, что за этим словом нужно еще что-то приписать.

Справка. Режим помощи программисту — это сердцевина среды программирования QBASIC. Как почти во всех программных продуктах, обращение к нему идет через нажатие клавиши-"выручалочки" F1. Но характер экранной подсказки будет зависеть от конкретного момента работы со средой QBASIC. Если, например, программист набрал какое-то служебное слово языка BASIC (READ, COLOR, PLAY и т.д.) , но забыл, какие параметры нужно за ним проставить, то после нажатия клавиши F1 на экране появится полное описание требуемого оператора и несколько типичных примеров его использования, один из которых можно будет перенести в текст программы, а затем отредактировать, приспособив к конкретному программному сюжету. Контекстная помощь среды QBASIC устроена так, что после нажатия F1 на экран выдается информация по служебному слову или по задействованной в программе переменной, функции, процедуре и т.д. Все зависит от того, что в данный момент в среде программирования «окурсорено». Например, программист обратился к меню среды QBASIC, пожелав записать готовую программу на диск и выбрав режимы «Файл» и «Сохранить как...», но при этом засомневался в правильности своих действий. Нажатие клавиши F1 выдаст на экран подробную инструкцию по обмену информацией с дисками. Подобный момент диалога программиста с машиной зафиксирован на рис.1 — курсор находится на первой строке восьмой позиции (см. нижний правый угол рис. 1 — № 00001:008) программы, поэтому за словом «Тема» в меню «Справка» записано REM. Если теперь программист нажмет на F1, то он узнает все об этом операторе. Оператор REM не такой уж и простой, как это может показаться. У него есть «псевдоним» — знак апострофа, перед которым двоеточие (знак-разделитель операторов на строке) ставится лишь для того, чтобы нейтрализовать действие оператора DATA (см. строки 2 — 5 на рис.1). Начинающим осваивать среду QBASIC адресованы позиции «Предметный указатель» и «Содержание» в меню «Справка»: с их помощью можно найти и просмотреть описание любого раздела среды QBASIC. Помощь у среды QBASIC, как убедился читатель, довольно сложная и для работы с ней программист сам нуждается в помо-

щи. Отсюда и появилась колонка меню «Справка». Команда «Использование справки», так сказать, суперсправка, обратиться за которой нужно также суперспособом — аккордом, нажав F1 и притопив клавишу Shift.

Заканчивая разговор о меню среды QBASIC и переходя к описанию собственно программирования на этом языке, отметим, что характер подачи на дисплее названий режимов сам содержит в себе важную подсказку. На рис. 1 все названия «блюд» вышли однотонными. На дисплее же разные слова прорисовываются либо в сочном, либо в затененном изображении. Сочность букв означает, что в данной ситуации этот режим выполним. Слабая прорисовка букв подсказывает программисту, что сейчас этим режимом пользоваться нельзя либо он просто бессмыслен (режим «Сохранить» или «Печать...» при чистом экране, например). Если же человек не послушается машины и ткнет все-таки в неработающий режим клавишей Enter, то машина в ответ пропищит, как бы моля о пощаде и отказываясь выполнять невыпопнимое или неразумное.

Теперь переходим к описанию самого программирования в таких комфортных условиях.

Среда QBASIC не смотрит свысока на своего предшественника, имя которого уже промелькнуло в статье — GW-BASIC. Если вы создали целую библиотеку программ на этом языке, то при переходе к работе со средой QBASIC она не пропадет. Это как при переходе к работе с более совершенной моделью калькулятора (от Б3-34 к МК-61), которая выполняет все команды своей предшественницы, но может делать еще что-то новенькое. Главное, чтобы тексты программ, написанные на языке GW-BASIC, хранились на магнитном диске в виде текстовых ASCII-файлов, а не в виде сжатой абракадабры, в которую GW-BASIC имеет обыкновение превращать программы при их записи на диск для последующего быстрого считывания и прогонки. GW-BASIC по команде SAVE «Имя программы» записывает программу в виде файла, текст которого человеку прочесть трудно, но который быстро загружается с диска по команде LOAD и быстро выполняется по команде RUN. Текстовым файл получится, если его переписать на диск по команде «SAVE »Имя программы",A" (A — это первая буква термина «ASCII-файл» — текстовый файл). Программы в виде текстового файла среда QBASIC подхватывает и выполняет. Возможны только небольшие заминки, связанные, например, с тем, что переменная IS на языке GW-BASIC в среде QBASIC не проходит, так как совпадает по имени со служебным словом IS ("есть" — см. далее рис. 9 и 10). В приложении читатель найдет перечень «подводных камней» на пути перевода GW-BASIC-программ для использования в среде QBASIC.

Вернемся к рис.1. Мы описали, чем окружены программы в среде QBASIC. Теперь настала пора рассказать о том, что находится в окружении слов-слуг среды программирования QBASIC. На рис. 1 можно увидеть самые короткие программы перевода арабских чисел в римские и наоборот, написанные на языке GW-BASIC (римские в арабские) и на языке QBASIC (арабские в римские; эта программа записана на рис. 1 комментариями строк GW_BASIC-программы). Транслятор GW-BASIC выполняет только первую программу, а транслятор QBASIC — обе, хотя их алгоритмы совершенно одинаковы. А почему так происходит, ответит табл. 1.

Таблица 1

Сравнительный анализ GW-BASIC и QBASIC

	GW-BASIC	QBASIC
	Необходимо	Не обязательно
Блоки DO WHILE/UNTIL и LOOP WHILE/UNTIL	Нет	Да
Блок IF/THEN/ELSE/END IF	Нет	Да
Блоки процедур (SUB и FUNCTION)	Нет	Да
Рекурсия	Нет	Да
Определяемые пользователем типы	Нет	Да
Блок SELECT CASE	Нет	Да
Длинные целые (32 бита)	Нет	Да
Номера формата IEEE	Нет	Да
Константы	Нет	Да
Строки фиксированной длины	Нет	Да
Режимы экрана VGA	Нет	Да
Поддержка видеокарт Hercules/Olivetti	Нет	Да
Поддержка кассетного магнитофона	Да	Нет
Общий размер программы (код + данные)	64 Кбайт	160 Кб
Единичный шаг	Нет	Да
Шаг процедуры	Нет	Да
Трассировка	Нет	Да
Контрольные точки	Нет	Да
Редактирование и продолжение	Нет	Да
Проверка синтаксиса	Нет	Да
Диагностические сообщения об ошибках	Нет	Да
Справка по ошибкам	Нет	Да
Несколько окон	Нет	Да
Поддержка мыши	Нет	Да
Интерфейс меню	Нет	Да
Совместимость с WordStar	Нет	Да
Организация кодов	Линейная	Модульная
Режим документов для редакт. текста	Нет	Да
Инструмент поиска и замены текста	Нет	Да
Встроенная справка	Нет	Полная

Сравнивая программы на рис. 1, можно отметить такие главные особенности языка QBASIC по сравнению с его предшественником GW-BASIC, отмеченные первыми строками табл.1.

Во-первых, строки операторов на языке QBASIC могут не нумероваться. Если перед введенной строкой на языке GW-BASIC не было номера, то операторы такой строки после нажатия клавиши Enter превращались в команды и сразу выполнялись. В среде программирования QBASIC безномерная строка, записываемая в окне редактирования, после нажатия клавиши Enter, просто заносится в программу, если, конечно, в ней нет ошибок и включена проверка ошибок (см. выше). На QBASIC для непосредственного выполнения команд в режиме калькулятора или в режиме отладки программ предусмотрено специальное окно (см. нижнюю часть рис. 1) с заголовком «Немед-

ленно». Верхнее же окно предназначено только для ввода и редактирования программных строк. Перекидывается курсор из одного окна в другое и обратно нажатием клавиши F6. У среды QBASIC есть и третье окно, куда по командам PRINT и по другим подобным заносятся результаты прогонки программы. Просмотреть это окно, как уже было отмечено, можно, выполнив в меню «Просмотр» команду «Экран вывода» или просто нажав F4. В пакете QBASIC есть специальная BASIC-программа с именем remline.bas, позволяющая из GW-BASIC-программ удалять все номера строк, не являющиеся метками. Тем не менее в языке QBASIC не отказались совсем от понятия «строка программы», как это сделано в языке Pascal, например. В QBASIC, как и в GW-BASIC, плечо альтернативы может начинаться со слова THEN и кончаться вместе со строкой. Отметим, глядя в программы на рис. 1, что в языке QBASIC сохранились суффиксы для выделения типа переменных: $ — строка. Кроме того, можно использовать и другие суффиксы: ! — обычная точность, % — целое, # — двойная точность и & — «длинное» целое. По умолчанию (QBASIC, как и GW-BASIC да и как любой другой BASIC, не стесняется в использовании этого довольно-таки «скользкого» программистского принципа) переменным без синтаксиса присваивается тип — обычная точность.

Во-вторых, в среде QBASIC можно обойтись не только без номеров строк, выполняющих по совместительству должность меток (GOTO 9, например), но даже и без операторов условного и безусловного перехода. Для разветвленных алгоритмов в среде QBASIC предусмотрены специальные структурные управляющие конструкции. О них мы сейчас и поговорим.

В программах — победительницах конкурса (см. рис.1) реализована основная структурная управляющая конструкция «цикл» в двух разновидностях: цикл «до» (итерация с постпроверкой — строки 7-13) и цикл «пока» (итерация с предпроверкой — строки 9-12). Козьма Прутков в одном из своих афоризмов вопрошал: «Что полезнее — солнце или месяц?» Сам же потом и отвечал: «Месяц. Ибо солнце светит днем, когда и без того светло, а месяц — ночью!» Подобным образом можно ответить и на вопрос: «Что важнее — цикл »пока" или цикл «до»?" Цикл «пока», конечно. Операторы тела цикла «до» выполняются как минимум один раз, что иногда излишне. Операторы же цикла «пока» в ряде случаев могут не работать ни разу, что должно отвечать задуманному алгоритму и мудрой пословице «Семь раз отмерь (сначала проверь условие завершения циклических действий), а потом отрежь (выполни тело цикла)». Синтаксис цикла в среде QBASIC запомнить проще простого: делай (DO) до тех пор, пока (UNTIL или WHILE) не LOOPнет чье-то терпение — пока не изменит свое значение логическое выражение, управляющее циклом.

Программа поиска корня уравнения методом половинного деления (рис. 2) содержит вторую управляющую конструкцию — полную альтернативу (IF...THEN...ELSE...END IF). То, что QBASIC может обходиться без номеров строк (см. рис. 1), пользователи языка GW-BASIC принимают не сразу, но в конце концов принимают. Особенность же среды QBASIC, реализованная в программе на рис. 2, часто их ставит в тупик и бывает причиной многих непонятных на первый взгляд ошибок. Дело в том, что переменные A и B в самой программе и в процедуре (SUB) с именем F расчета

```
DECLARE SUB F (X, Y)'См.процедуру внизу
INPUT «A,   B,Точность »; A, B, Eps
F A, Y1'   Расчет Y на левом конце отрезка
DO WHILE B — A > Eps '      Начало цикла
   X = (A + B) / 2 '        Отрезок пополам
   CALL F(X, Y)'          Расчет Y в середине
   IF Y1 * Y >= 0 THEN '     Альтернатива
      A = X '                 Левое плечо
            ELSE B = X '      Правое плечо
   END IF '                Конец альтернативы
LOOP '                        Конец цикла
PRINT «Y = 0 при X = »; X '          Ответ

SUB F (Arg, Func) '       Заголовок процедуры
  A = Arg ' Пересылка в локальных переменных
  B = A * A '     Вторая локальная переменная
  Func = B — 3 ' F(X)=X*X-3
END SUB '                      Конец процедуры
```

Рис. 2. «Дейкстра любит троицу». В программе поиска корня алгебраического уравнения методом половинного деления мы видим три основные алгоритмические конструкции: следование (оператор за оператором), повторение (DO...LOOP) и выбор (IF...THEN...ELSE...END IF)

значения анализируемой функции разные и независимые. Это новшество , приближающее QBASIC к таким почтенным языкам, как QuickBASIC, Turbo BASIC, Pascal, C, fortran, PL/1, позволяет программисту, во-первых, поручать написание отдельных процедур (функций) своим подмастерьям, не заботясь при этом о разделении переменных, и во-вторых, дает возможность быстро компоновать программу из отдельных готовых и отлаженных модулей. В программе на рис. 2 вызов процедуры ведется двумя способами: по-паскалевски (см. третью строку) и по-фортрановски (см. вторую строку тела цикла). Язык BASIC, как известно, характеризуется многовариантностью, когда одну и ту же программистскую «мысль» (вызов процедуры, как на рис. 2, объявление типа переменной и т.д.) можно высказать разными языковыми средствами.

Разделение переменных, неизбежное при работе с языком GW-BASIC, в среде QBASIC машина берет на себя. «Начальнику программы», работая по такой технологии «сверху вниз», можно даже не сообщать помощникам имена входных (у нас это переменная X в программе и Arg в процедуре) и выходных (переменная Y в программе и Func в процедуре) параметров. А какие локальные переменные задействуют «подмастерья» — это зависит от их вкуса и привычек. С глобальными переменными и переменными других подпрограмм они не смешаются и алгоритм решения задачи тем самым не сломают: «Вассал моего вассала не мой вассал» — девиз феодальной эпохи в истории человечества. «Переменная моей процедуры — не моя переменная» — девиз эпохи структурного программирования. «Всеобщая уравниловка» в рангах переменных, процедур, функций — вот одна из «ахиллесовых пят» языка GW-BASIC, куда пускали критические стрелы приверженцы языков C, Pascal, fortran.

Две последующие программы (на языке QBASIC — см. рис. 3 и на языке GW-BASIC - см. рис. 4) поиска минимума функции методом золо-

```
DEF FNY (X) = X * X – N
INPUT «A, B, Точность»; A, B, Eps
CONST N = 3
GOSUB RIGHT: GOSUB LEFT
DO WHILE ABS(B – A) > Eps
   Flag = 1
   DO WHILE Y1 < Y2 AND Flag
     B = X2: X2 = X1: Y2 = Y1
     GOSUB RIGHT  :  Flag = 0
   LOOP
   DO WHILE Flag
     A = X1: X1 = X2: Y1 = Y2
     GOSUB LEFT  :  Flag = 0
   LOOP
LOOP
PRINT «Y MIN=»; FNY(X); « при X=»; X
STOP

RIGHT:
  X1 = .618 * A + .382 * B: Y1 = FNY(X1)
RETURN
LEFT:
  X2 = .382 * A + .618 * B: Y2 = FNY(X2)
RETURN
```

Рис. 3. Попросите знакомого художника разделить отрезок на две неравные части. Можно быть уверенным, что он сделает это в золотом соотношении. Подобное чувство меры позволяет компьютеру так делить интервал неопределенности A-B, что при очередном приближении к минимуму высчитывается только одно значение функции. См. также подпись под рис. 5

того сечения отвечают на вопрос о том, какие структурные управляющие конструкции следует считать основными, а какие вспомогательными, без которых можно обойтись. Основная структурная теорема (ее автор американский математик и программист Дейкстра) на этот вопрос отвечает так: «Алгоритм любой сложности можно реализовать, используя триаду »следование – повторение (циклы) – выбор (альтернатива)". Но программы на рис. 3 и 4 доказывают, что при особом желании можно обойтись и без альтернативы: в GW-BASIC-программе (см. рис. 4) она есть, а в ее аналоге на языке QBASIC нет (см. рис. 3). Программе на рис. 3 хватило цикла «пока» и переменной Flag (вещественной по форме и булевой по содержанию), заставляющей операторы тела циклов работать либо раз, либо совсем ни разу. Раз речь зашла о булевых переменных, то отметим, что QBASIC имеет полный набор булевых операций: NOT – отрицание, AND – конъюнкция,OR – дизъюнкция (включающее «или»), XOR – исключающее «или», EQV – эквивалентность иIMP – импликация.

Каждая операция возвращает результат, как это приведено в табл. 2: И – «истина» (не-ноль, в среде QBASIC минус единица) и Л – ложь (ноль).

Таблица 2

Выра жение1	Выра жение2	NOT	AND	OR	XOR	EQV	IMP
И	И	Л	И	И	Л	И	И
И	Л	Л	Л	И	И	Л	Л
Л	И	И	Л	И	И	Л	И
Л	Л	И	Л	Л	Л	И	И

В языке QBASIC, как видно из рис.3, осталось понятие метки, которая является не только номером строки, но и может иметь собственное имя (RIGHT, LEFT). Остались и традиционные BASICовские эрзац-подпрограммы с операторами GOSUB и RETURN. Так что приверженцы языка GW-BASIC, перейдя к работе с языком QBASIC, могут использовать милые их сердцу приемы. В программе на рис.3 оператором CONST переменная N зафиксирована как константа.

Козьма Прутков, развивая свою мысль о солнце и месяце (см. выше), в следующем афоризме утверждал: «Но, с другой стороны: солнце лучше тем, что светит и греет; а месяц только светит, и то лишь в лунную ночь!» Продолжим подобные рассуждения, перенеся их на понятия цикл «пока» и цикл «до». В QBASIC-программе поиска корня алгебраического уравнения методом Ньютона (рис. 5) использован еще один цикл – цикл с выходом из середины. Вот эту-то конструкцию, а не цикл «пока»и не цикл «до», по-видимому, и следует считать основной, без которой не реализуешь задуманный алгоритм. Незаменимость цикла с выходом из середины заключается в том, что выходов из цикла может быть несколько. Этого не избежать, когда значение логического выражения, управляющего циклом меняется вне цикла, и есть опасение пропустить в нужный момент ее изменение со значения «да» на значение «нет» (или наоборот). Такое случается тогда, когда в программе задействованы элементы параллельного программирования, когда, например, управляющее циклом логическое выражение опирается на показание встроенных часов компьютера или на опрос состояния периферии (включен ли принтер, вставлена ли в него бумага, есть ли в дисководе диск, отформатирован ли он и т.д. и т.п.). Отметим, что цикл с выходом из середины на языке Pascal реализуется либо с опорой на метку (а это признак низкой культуры программирования), либо через организацию процедуры. Обсуждая далее довольно спорный вопрос о необходимом и достаточном наборе управляющих конструкций языка, отметим, что многие программисты считают основной и незаменимой

```
1 DEF FNY (X) = X * X – 3
2 INPUT «A,B,Точность»;A,B,Eps: GOSUB          8:GOSUB 9
3: IF ABS(B – A) <= Eps GOTO 7
4 IF Y1<Y2 THEN B=X2:X2=X1:Y2=Y1:GOSUB 8:GOTO 6
5 A = X1:  X1 = X2:  Y1 = Y2:    GOSUB 9
6: GOTO 3
7: PRINT «Y MIN=»; FNY(X); « при X=»; X: END
8: X1 = .618 * A + .382 * B: Y1 = FNY(X1):RETURN
9: X2 = .382 * A + .618 * B: Y2 = FNY(X2):RETURN
```

Рис. 4. Подчеркнуть вторую роль номеров строк в GW-BASIC-программах можно знаком «двоеточие» перед операторами, к которым ведется переход через GOTO, GOSUB и другие операторы, превращающие тексты GW-BASIC-программ в «спагетти»

```
DECLARE FUNCTION Y! (X!)
DECLARE FUNCTION D! (X!)
DIM SHARED Constant
INPUT «X начальн., точность»; X, Eps
Constant = 3
DO ' Начало цикла
   X1 = X - Y(X) / D(X)
 IF ABS(X1 - X) < Eps THEN EXIT DO
   X = X1
LOOP ' Конец цикла
PRINT «Y=0  при X =»; X

FUNCTION D (X)
   D = 2 * X
END FUNCTION

FUNCTION Y (X)
   Y = X * X - Constant
END FUNCTION
```

Рис. 5. Для функции пользователя, объявленной через DEF FNимя, доступны все переменные основной программы, включая и переменную N (см. рис.3). Барьер операторов DECLARE FUNCTION и DECLARE SUB языка QBASIC осилят лишь те переменные, каким оператор DIM SHARED присвоил «высокий дипломатический ранг», как в программе на этом рисунке

конструкцией, помимо цикла и альтернативы, также и рекурсивный вызов процедуры или функции, когда процедура или функция «лезет поперек батьки в пекло» — вызывает сама себя еще до того, как ее до конца определят. На старых версиях языка BASIC рекурсия использовалась по принципу «Если нельзя, но очень хочется, то можно» (см., например, журнал «Наука и жизнь» № 12 за 1988 г., стр. 88-89 с описанием рекурсивного решения задачи о ханойских башнях на микроЭВМ «Искра 226» — WANG-2200). В среде QBASIC рекурсия разрешена (см. программу «Ханойские башни» на рис. 6).

```
DEFINT N ' Все переменные на N целые
DEFSTR X-Z ' Остальные — литерные
DECLARE SUB PUTDISK (N%,X$,Y$,Z$)
INPUT «Число дисков»; N0
PUTDISK N0, «A», «B», «C»
END

SUB PUTDISK (N%,X$,Y$,Z$)
DEFINT N: DEFSTR X-Z
IF N = 1 THEN
          PRINT X; «-»; Z
        ELSE
     PUTDISK N - 1, X, Z, Y
     PRINT X; «-»; Z
     PUTDISK N - 1, Y, X, Z
   END IF
END SUB
```

Рис. 6. Задача о ханойских башнях кочует из одного учебника по языкам программирования в другой и вот попала в нашу статью для иллюстрации рекурсии на языке QBASIC

Еще раз подчеркнем многовариантность языка QBASIC, отметив, что на этом языке тип переменной может задаваться либо по умолчанию (вещественная одинарной точности), либо суффиксом (!, #, $, % или &), либо по-Фортрановски, как в первой строке программы на рис. 6 (правда, на языке Fortran переменная N целочисленная по умолчанию). Как и в языке Pascal, в языке QBASIC есть оператор с ключевыми словами для задания типа переменной: DIM переменная AS INTEGER — 16-битовая целая переменная со знаком (от 32767 до -32768), LONG — 32-битовая целая переменная со знаком (от 2147483647 до -2147483648), SINGLE — 32-битовая переменная одинарной точности с плавающей запятой (положительные от 3.402823E+38 до 2.802597E-45, отрицательные от -2.802597E-45 до -3.402823E+38), DOUBLE — 64-битовая переменная двойной точности с плавающей запятой (положительные от 1.79769313486231D+308 до 4.940656458412465D-324, отрицательные от -4.940656458412465D-324 до -1.79769313486231D+308), STRING * n% строковая переменная фиксированной длины (длина — n% байтов) и STRING — строковая переменная переменной длины.

В описание задачи о ханойских башнях относительно рекурсии отметим только, что при ее решении нужно перенести диски пирамиды (первая ханойская башня) со стержня A на стержень C (третья ханойская башня), беря их по одному и пользуясь промежуточным стержнем B. При этом нельзя большой диск класть на маленький. На исходном стержне диски лежат в таком порядке: самый крупный — внизу, а самый маленький — наверху. Задача с двумя дисками решается просто: с A на B, с A на C и, наконец, с B на C. При трех дисках последовательность действий такая: A — C, A — B, C — B, A — C, B — A, B — C и A — C. Число действий в общем случае равно 2 в степени N-1, где N — число дисков. Вдумчивый читатель может отметить, что задачу о перестановке N дисков можно упростить, сведя ее к серии задач о перестановке N-1 дисков. Отсюда и рекурсия в программе на рис. 6.

Текст QBASIC-программы, кстати, несложно переделать для Pascal-транслятора, раз мы упомянули этот язык абзацем выше. Более менее опытный программист справится с такой работой в два счета, используя такие функции редактора, как возможность перестановки местами строк и целых блоков текста, а также замены одних символов другими. Нужно будет, например, заменить в операторах присвоения знак «равно» (=) на знак «соответствует» (:=), оставив знак «равно» только в логических выражениях. Символ «=» в BASIC-программах несет двойную нагрузку, и оператор A=B=C имеет вполне определенный смысл: переменной A будет возвращено значение «минус единица» (бейсиковская истина), если B равно C, или «ноль» (бейсиковская ложь) в противном случае. В Pascal-программе в аналогичной ситуации нужно будет записать A:=B=C и «сбегать» в начало программы, чтобы сообщить транслятору, что A — это булевая переменная. QBASIC-интерпретатор в некоторых ситуациях сам «бегает» к началу программы, например для того, чтобы декларировать функции и процедуры, а также объявлять тип их входных и выходных переменных-параметров. Так операторы DECLARE SUB (см. рис. 2) и DECLARE FUNCTION (см. рис. 5) были в программе приписаны интерпретатором перед записью их на диск.

В популярной математической среде Eureka фирмы Borland оба символа ("=", «:=») являются знаками присвоения, но имеют в этой операции разный приоритет и несколько разные особенности. В другой популярной математической среде MathCAD фирмы MathSoft символ «=» эквивалентен Бейсиковскому оператору PRINT, а символ «:=» — INPUT. С языками программирования не соскучишься! Для скучающих (неужели такие найдутся среди читателей!) приводим список спецсимволов языка QBASIC:

«*» — знак умножения;

«-» знак минуса;

«/» — знак деления (слэш);

«=» — оператор равенства или символ присвоения;

«>» знак больше, чем (что!);

«+» знак плюса; «.» (или «,») — десятичная запятая;

«<» — знак меньше, чем (что!);

«\» — знак целочисленного деления (обратный слэш);

«^» — знак показателя степени;

«'» — строка комментария (одиночные кавычки);

«;» и «,» — управляющие символы для операторов PRINT и INPUT;

«:» — разделитель нескольких операторов в одной строке;

«_» — символ подчеркивания (сохраняется для совместимости с другими версиями BASIC, не поддерживается QBASIC);

«?» — приглашение при операторе INPUT (кроме того, заменяет ключевое слово PRINT).

Строка в Pascal-программе в отличие от BASIC-строки не несет никакой смысловой нагрузки и ее можно рвать как угодно. Функцию нажатия клавиши Enter в языке BASIC (или знака «двоеточие») несут слово End языка Pascal и знак «точка с запятой». Комментарии в Pascal-программах заключаются в фигурные скобки, чтобы машина заранее знала, какую «воду» нужно будет «выжимать» из программы при ее компиляции. Попутно заметим, что фигурные скобки в С-программах имеют прямо противоположный смысл, отмечая не «воду», а «соль» программы: «{» — начало (в языке Pascal для этого пишется Begin) и «}» — конец (End) блока операторов. Напоминаем еще раз: с чем-чем, а с языками программирования не соскучишься!

На языке Pascal цикл с выходом из середины (см. рис. 5) так просто не запишешь. При необходимости он реализуется двумя способами. Во-первых, прервать цикл в любом месте можно с помощью метки и оператора условного перехода (If ... Goto ...). Те же, на кого метка в программе действует как красная тряпка на быка, поступают, как говорил Даниил Хармс, «самым остроумным способом», а именно: тело цикла записывается процедурой с вкраплениями If... Then Exit — т.е. с командами «Если... , то прерви выполнение процедуры и вернись в основную программу к оператору, записанному за именем процедуры». На языке QBASIC цикл можно прервать в любом месте любое число раз (EXIT DO — см. рис. 5 или EXIT FOR). Также бесцеремонно QBASIC позволяет поступать и с процедурами (EXIT SUB), и с функциями (EXIT FUNCTOIN). На языке С пошли еще дальше: цикл можно прервать, передав управление программой либо в его конец (If...Then Break), либо в его начало (If...Then Continue). Так что «можно быть более католиком, чем папа римский» и более структурированным языком, чем Pascal.

Если читатель еще не совсем запутался, то вот какую «ясность» в проблему минимального числа управляющих конструкций алгоритмов (суть основной структурной теоремы Дейкстры) внесла переведенная с английского и выпущенная издательством «Мир» в 1989 г. книга «Языки программирования Ада, Си и Паскаль. Сравнение и оценка». В ней на с. 72 и 73 сказано: «В соответствии с генеалогией управляющих структур <...> циклы с использованием оператора завершения Break и оператора продолжения Continue относятся к классу DREC(1). Теоремы, доказанные Р.Косараю, показывают, что управляющие структуры, относящиеся к классу DREC(1), не могут быть эмулированы с помощью управляющих структур, относящихся к классу D (название этого класса образовано от первой буквы фамилии Э.Дейкстры — автора основной структурной теоремы), которые составляются из произвольного числа условных операторов If, операторов цикла While и их конкатенаций. На самом деле управляющие структуры, относящиеся к классу DREC(i) (имеющие в своем составе оператор выхода Exit(i) (или Break), обеспечивающий выход на i уровней вверх, и оператор Cycle(i) (или Continue), обеспечивающий выполнение цикла на i-м уровне вложенности, как, например, в языке Блисс, являются более мощными, чем управляющие структуры, относящиеся к классу DREC(i-1) <...>. Однако программы, содержащие циклы с использованием операторов Exit и Cycle, оказываются намного сложнее для понимания. Анализ показывает, что необходимость использования оператора Exit возникает крайне редко. Необходимость наличия управляющих структур более высокого уровня, чем управляющие структуры класса D, до сих пор не доказана» (конец цитаты). Вот так-то. Начали «за упокой» основной структурной теоремы, а кончили «за здравие».

Конструкции DO ... IF ... THEN EXIT DO ... LOOP и FOR ... IF ... THEN EXIT FOR ... NEXT языка QBASIC придают ему еще большую гибкость за счет того, что между ключевыми словами THEN и EXIT можно вставлять операторы, фиксирующие место выхода из цикла.

В статью помещен табель-календарь грядущего года, составленный с учетом тематики статьи. Это QBASIC-программа распечатки табеля-календаря (рис. 7 — вариация на тему «цикл с параметром» — еще одна структурная управляющая конструкция). Годится ли программа для любого года и любого тысячелетия, нет ли в ней ошибок (для 1992 года ее точно нет) и как их исправить — вот задание для наших читателей. Подобные программы называются программами вечного календаря, наверно, и потому, что эта тема вечная: поток писем с подобными программами, поступающими в редакции компьютерных журналов, не иссякает.

В программе на рис. 7 использован массив переменных D. Приводим характеристики массивов в среде QBASIC. Размер массива (все элементы): статический — от 65535 байтов (max) до 1 байт (min); динамический — от 65535 байтов (max) до 0 байт (min). Число допустимых размерностей — от 60 (max) до 1 (min). Допустимые размерности по умолчанию 8 (max) до 1 (min). Значение индекса массива: от 32767 до -327 (замечание: максимальный диапазон между значениями индекса массива — 32767). По умолчанию (без использования оператора DIM как в программе на рис. 7) максимальный индекс массива 10.

```
PRINT «                Какое, милые, у нас      »
INPUT «                Тысячелетье на дворе ?     », B.Pasternak
DATA Пн.    , Вт.    , Ср.    , Чт.    , Пн.    , Сб.    , Вс.
DATA   ЯНВАРЬ  , 31 , ФЕВРАЛЬ , 28 ,   МАРТ  , 31 , АПРЕЛЬ  , 30
DATA    МАЙ   , 31 , ИЮНЬ  , 30 ,   ИЮЛЬ  , 31 , АВГУСТ  , 31
DATA СЕНТЯБРЬ , 30 , ОКТЯБРЬ , 31 , НОЯБРЬ , 30 , ДЕКАБРЬ , 31
Year = 1000 * (B.Pasternak - 1) + 993: CLS
PRINT «Табель-календарь на»; Year;
Day = INT(365.25 * (Year - INT(Year / 100) * 100 - 1)) + 429
D(5) = Day - INT(Day / 7) * 7 + 1
FOR C = 1 TO 7: READ D$(C): NEXT
FOR M = 1 TO 12 STEP 4
  D(1) = D(5)
  FOR I = 1 TO 4
    READ M$(I), K(I)
    PRINT TAB(18 * I - 11 + INT((17 - LEN(M$(I))) / 2)); M$(I);
    IF M < 2 AND Year / 4 = INT(Year / 4) THEN K(2) = 29
    IF D(I) <= 1 THEN D(I) = 8
    IF D(I) >= 9 THEN D(I) = D(I) - 7
    D(I + 1) = D(I) + K(I) - 28
  NEXT: PRINT
  FOR C = 1 TO 7
    PRINT D$(C); TAB(6);
    FOR I = 1 TO 4
      PRINT TAB(18 * I - 11);
      J0 = C - D(I) + 2
      IF J0 <= 0 THEN J0 = J0 + 7
      IF D(I) - 1 > C THEN PRINT «   »;
      FOR J = J0 TO K(I) STEP 7: PRINT USING «## »; J; : NEXT
    NEXT: PRINT
  NEXT
    NEXT
```

Рис. 7. Программу печати табеля-календаря можно дополнить операторами, выделяющими красным цветом или жирным шрифтом памятные даты: праздники, дни рождений, юбилей. Такой индивидуальный табель-календарь — отличный новогодний подарок

Еще одна неиссякаемая тема компьютерных журналов — это минимизация программ, печатающих сами себя. Вторую жизнь и вторую молодость дает ей QBASIC, и вот почему.

Колумбу история приписывает два великих дела: во-первых, открытие Америки, а во-вторых, решение задачи о «колумбовом» яйце. Ее суть в том, что яйцо требовалось поставить на острый конец на гладкой поверхности. Многие пытались для этого раскручивать яйца, делать незаметные подпорки. Колумб же слегка надломил скорлупу и поставил яйцо торчком.

Языку BASIC, как и Колумбу, история тоже приписывает два великих дела.

Во-первых, BASIC открыл для многих людей прекрасный мир программирования, а во-вторых, он по-колумбовски решил задачу о самопечати программ. Так, на языке PL/1 эта задача решается программой с полусотней операторов, а на языке GW-BASIC любая программа может быть распечатана одним-единственным оператором LIST как в непосредственном, так и в программном режиме. Но оператора LIST в среде QBASIC нет (есть ключевое слово LIST, но это не оператор и не команда), поэтому «жив курилка!»: вот вам, уважаемый читатель, «затравочный кристалл» — QBASIC-программа самопечати на рис. 8 (автор А.Сибирцев из г.Канска).

Сможете ли вы написать более короткую QBASIC-программу или учесть ограничения, отмеченные в подписи к рис. 8? Есть и другое задание. Сможете ли вы сократить программы перевода римских и арабских чисел (рис. 1) или написать другие, более короткие. Первое решение лежит на поверхности, в строке 6, где можно убрать операторы N=0 и N$="". Это еще раз подчеркивает, что QBASIC наследовал от языка GW-BASIC такое «родимое пятно», как необязательность объявления переменных и присваивание им по умолчанию нулевого значения.

А вот более практичные QBASIC-программы на рис. 9 и 10, решающие задачу о подоходном налоге с граждан. Эти программы можно считать вариациями на тему «Select — Выбор». Это еще одна управляющая конструкция алгоритмов, заменяющая серию альтернатив.

Читателям, успешно осваивающим персональные компьютеры и повышающим вследствие этого не только свой образовательный, но и материальный уровень, интересно вникнуть в задачу о налогах и сравнить, как это делается у «нас» (см. рис. 9) и как у «них» (см.

```
FOR I = 1 TO 9
READ A$
PRINT A$
NEXT
RESTORE
FOR I = 1 TO 9
READ A$
PRINT «DATA »; A$
NEXT
DATA FOR I = 1 TO 9
DATA READ A$
DATA PRINT A$
DATA NEXT
DATA RESTORE
DATA FOR I = 1 TO 9
DATA READ A$
DATA PRINT «DATA »; A$
DATA NEXT
```

Рис. 8. Часто в задаче о самопечати программ запрещается использование операторов DATA, REED и RESTORE и операторов обмена информацией с дисками. В нашей программе эти табу не принимались во внимание

рис. 10). Что при этом бросается в глаза? Во-первых, налоговая система США нацелена на укрепление семьи — если супруги захотят развестись, то государство сразу накажет их рублем, точнее, долларом. Во-вторых, в США никому в голову не пришло «драть» с налогоплательщика 60 и более процентов с заработка, превышающего определенную сумму. 33 процента — вот тот предел, на который может отважиться государство, не опасаясь подавить творческую активность своих кормильцев. В-третьих и главных, при высоких доходах ставка налога снижается до 28 процентов, так как преуспевающие налогоплательщики свои сверхдоходы в рыночной среде никогда не будут проедать, а пустят их в дело: купят акции, средства производства (тот же персональный компьютер или новую версию MS-DOS 5.0, к примеру), займутся благотворительностью и т.д.

Если «Календарь» (рис. 7) или «Самопечать» (рис. 8) можно считать вечной темой программистов, то проблему налогов можно считать вечной темой наших парламентариев. Программу на рис. 9 читатель может сравнить не только с программой на рис. 10 (финансовая сторона проблемы), но и с рис. 11 (программистская сторона проблемы).

```
DECLARE FUNCTION Pay! (ch!)
DECLARE FUNCTION Tax! (adjPay!)
INPUT « Чистые »; ch
PRINT USING « Грязные = ######.##»; Pay(ch)
PRINT USING « Налог = #####.##»; Tax(Pay(ch))

FUNCTION Pay (ch)
   tmp1 = ch
   DO
        tmp2 = ch + Tax(tmp1)
      IF ABS(tmp1 - tmp2) < .001 THEN EXIT DO
        tmp1 = tmp2
   LOOP
   Pay = tmp2
END FUNCTION

FUNCTION Tax (adjPay)
a = adjPay
SELECT CASE a
 CASE IS < 3500:    T = .12 * a
 CASE 3500 TO 7000: T = 420 + .15 * (a - 3500)
 CASE 7000 TO 10000'          «А то что ж: один в семи комнатах
 T = 945 + .2 * (a - 7000)'   расселился, штанов у него 40 пар,
 CASE 10000 TO 15000' а другой шляется, в сорных ящиках питание
 T = 1545 + .3 * (a - 10000)'                          ищет",
 CASE 15000 TO 25000'                                 ремарка
 T = 3045 + .4 * (a - 15000)' Полиграфа Полиграфовича Шарикова.
 CASE 25000 TO 35000 '        Они спутали заработанную плату со
 T = 7045 + .5 * (a - 25000)'          сверхприбылью монополии.
 CASE IS > 35000'          Комментарий, не пропущенный в печать
 T = 12045 + .6 * (a - 35000)' внутренним цензором программиста
END SELECT
```

Рис. 9. Локальные переменные с именами «а» (вместо глобальной adjPay) и «Т» (вместо Tax) делают «налоговые» функции более легкими и понятными. Вот где пригодилось нам знакомство с методами поиска корней алгебраических уравнений

Литература

1. *Очков В.Ф., Пухначев Ю.В.* 24 этюда на Бейсике. — М.: Финансы и статистика, 1988
2. *Очков В.Ф., Хмелюк В.А.* От микрокалькулятора к персональному компьютеру. — М.: Изд-во МЭИ, 1990.
3. Школа пользователя ЭВМ:/В.Птицын (Очков В.Ф.): Среда программирования QuickBASIC 4.5 // Наука и жизнь. — 1990. — № 11,
4. *Очков В.Ф., Пухначев Ю.В.* 128 советов начинающему программисту. М.: Энергоатомиздат, 1991.
5. *Очков В.Ф.* Языки GW-BASIC и QBASIC: сравнительное описание. — М.: Издательская фирма «Логос» (В печати).
6. *Halvorson M., Rygmyr D.* Running QBasic. — Microsoft Press, 1991.
7. *Jamsa K.* QBASIC: Programmer's Quick Reference. — Microsoft Press, 1991.
8. «The Waite Group's QBASIC Reference Plus» by Waite, Prata, et al. — Microsoft Press, 1991.
9. «The Waite Group's Microsoft QuickBASIC Bible» by Waite, Prata, et al. — Microsoft Press, 1990.
10. «The Waite Group's Microsoft QuickBASIC Primer Plus» by Waite et al. — Microsoft Press, 1990.
11. «Microsoft QuickBASIC Programmer's Toolbox» by John Clark Craig. — Microsoft Press; 1988.
12. «Microsoft QuickBASIC» by Douglas Hergert (third edition. — Microsoft Press, 1988.
13. *David I. Schneider.* Microsoft QuickBASIC: An Introduction to Structured Programming. — Dellen Publishing Company. — San Francisco, 1989.

Ключевые слова языка QBASIC

CALL ABSOLUTE ([аргументы,] смещение%). Передает управление в процедуру на языке Ассемблер. Аргументы, переданные в процедуру на языке Ассемблер такие, как смещения относительно текущего сегмента данных. Смещение от текущего кода сегмента, установленного DEF SEG, до выполнения процедуры.

ACCESS {READ | WRITE | READ WRITE}. Указывает тип доступа, который получает пользователь к открытому файлу. READ открывает файл только для чтения. WRITE открывает файл только для записи. READ WRITE открывает файл как для чтения, так и для записи. Режим READ WRITE дей-

```
DECLARE FUNCTION TaxMarried (adjPay)
DECLARE FUNCTION TaxSingle (adjPay)
INPUT «Грязные ($ в неделю) »; adjPay
PRINT «Налог с холостого (незамужней) = »; TaxSingle(adjPay); «$»
PRINT «Налог с женатого (замужней) = »; TaxMarried(adjPay); «$»
END

FUNCTION TaxMarried (adjPay)
 'Расчет федерального налога с женатого (замужней)
 a = adjPay
 SELECT CASE a' Заголовок конструкции «Выбор»
   CASE 0 TO 62: T = 0
   CASE 62 TO 657: T = .15 * (a – 62)
   CASE 657 TO 1501: T = 89.25 + .28 * (a – 657)
   CASE 1501 TO 3695: T = 325.57 + .33 * (a – 1501)
   CASE IS >= 3695: T = 1049.59 + .28 * (a – 3695)
 END SELECT' Конец конструкции «Выбор»
TaxMarried = T
END FUNCTION

FUNCTION TaxSingle (adjPay)
 'Расчет федерального налога с холостого (незамужней)
 a = adjPay
 SELECT CASE a
   CASE 0 TO 21: T = 0
   CASE 21 TO 378: T = .15 * (a – 21)
   CASE 378 TO 885: T = 53.55 + .28 * (a – 378)
   CASE 885 TO 2028: T = 195.51 + .33 * (a – 885)
   CASE IS >= 2028: T = 572.7 + .28 * (a – 2028)
 END SELECT
TaxSingle = T
END FUNCTION
```

Рис. 10. «Некруглость» чисел в формулах не допускает скачков «налоговых» функций

ствителен только для файлов прямого доступа и бинарных, а также для файлов, открытых с APPEND (последовательный доступ).

ANY. Используется в операторе DECLARE FUNCTION для описания типа переменных. ANY допускает использование любого типа данных.

APPEND. В операторе OPEN указывает, что файл должен быть открыт для последовательного вывода, и устанавливает указатель файла в конец файла. В этом случае операторы PRINT# или WRITE# дополняют файл.

AS. Выполняет разные функции как часть различных операторов: указывает тип переменной при использовании с операторами **COMMON, DECLARE, DEF FN, DIM, FUNCTION, REDIM, SHARED, STATIC и SUB**; указывает тип элемента в типе данных, определенных пользователем, при использовании с оператором TYPE: присваивает номер файла файлу или устройству при использовании с оператором OPEN; указывает имя поля при использовании с оператором FIELD; указывает новое имя файла при использовании с оператором NAME.

BASE. См. описание оператора OPTION BASE

BINARY. Используется в операторах OPEN и OPEN COM и указывает бинарный режим файла. В бинарном режиме можно считывать или записывать информацию в позицию любого байта, используя операторы GET и PUT.

CASE. Используется в операторе SELECT CASE

DOUBLE. Указывает тип данных для переменных в операторах описания и в списке параметров: 64-битовая переменная двойной точности с плавающейзапятой.

ELSE и ELSE IF. Используются в операторе IF ... THEN ... ELSE

INTEGER. Указывает тип данных для переменных в операторах описания и в списке параметров: 16-битовая целая переменная со знаком.

IS. Используется в операторе SELECT CASE

LIST. Используется в операторе KEY (присваивание). KEY LIST отображает установки для каждой клавиши.

LONG. Указывает тип данных для переменных в операторах описания и в списке параметров: 32-битовая целая переменная со знаком.

LOOP. Используется в операторе DO...LOOP

NEXT. Осуществляет приращение и проверку счетчика в цикле FOR...NEXT или при использовании с RESUME продолжает выполнение, начиная с программыотслеживания ошибок.

```
10  DEF FNnal (g)
20   IF g < 210 THEN FNnal = 0: GOTO 70
30   IF g < 1000 THEN FNnal = .12 * (g – 210): GOTO 70
40   IF g < 2000 THEN FNnal = 94.8 + .15 * (g – 1000): GOTO 70
50   IF g < 3000 THEN FNnal = 244.8 + .2 * (g – 2000): GOTO 70
60   FNnal = 444.8 + .3 * (g – 3000)
70  END DEF
80  INPUT «Грязные »; g
90  PRINT USING « Чистые = #####.##»; g – FNnal(g)
100 PRINT USING « Налог = #####.##»; FNnal(g)
```

Рис. 11. В GW-BASIC-программе вычисления налогов по щадящим процентам, действовавшим, когда союзные структуры переманивались в российские, задействованы операторы условных и безусловных переходов к меткам. Такой прием программирования превращал программы в «спагетти», из которой очень трудно «выудить» отдельный оператор или строку

OFF. Выключает отображение присвоений функциональных клавиш при использовании с оператором KEY (присваивание) или выключает отслеживание события при использовании операторами события OFF (COM OFF, KEY OFF, PEN OFF, PLAY OFF, STRIG OFF и TIMER OFF)

ON. Выполняет различные действия в качестве части некоторых операторов: включает отслеживание ошибки при использовании с оператором ON ERROR; включает отслеживание события при использовании с операторами события ON (COM ON, KEY ON, PEN ON, PLAY ON, STRIG ON и TIMER ON); указывает, какое событие отслеживать при использовании с ON-операторами события (ON COM, ON KEY, ON PEN, ON PLAY, ON STRING, and ON TIMER); указывает, какое выражение оценивать при использовании с операторами ON...GOSUB и ON...GOTO

OUTPUT. Используется в операторах OPEN и OPEN COM, где указывает, что файл открыт для последовательного вывода.

RANDOM. Используется в операторах OPEN и OPEN COM и указывает, что файл открыт в режиме прямого доступа к файлу. Этот режим устанавливается по умолчанию.

SINGLE. Указывает тип данных для переменных в операторах описания и в списке параметров: 32-битовая переменная одинарной точности с плавающей запятой.

STEP. В цикле FOR...NEXT указывает, насколько увеличивать значение счетчика при каждой итерации. В графических операторах указывает, что координаты точки растра заданы относительно текущего графического положения курсора.

STRING. Указывает тип данных для переменных в операторах описания и в списке параметров: STRING * n%, где n% — строковая переменная фиксированной длины (длина — n% байтов). STRING строковая переменная переменной длины.

TO. Определяет диапазон для: предложения CASE в операторе SELECT CASE; счетчика цикла в цикле FOR...NEXT; закрываемых и открываемых записей в операторе LOCK...UNLOCK; верхней и нижней границы в операторе DIM или REDIM.

UNTIL. Используется в операторе DO...LOOP

USING. Указывает форматирование для операторов PRINT USING и LPRINT USING, а также назначения палитры в операторе PALETTE USING

WEND. Используется в операторе WHILE...WEND

Функции языка QBASIC

ABS (числовое_выражение). Возвращает абсолютное значение числа.

ASC (строк_выражение$). Возвращает ASCII код первого символа в строковом выражении.

ATN (числ_выражение). Возвращает арктангенс указанного числового выражения (угол в радианах).

CDBI. (числ_выражение) переводит числовое выражение в значение с двойной точностью.

CHR$(ascii-код%). Возвращает символ, соответствующий указанному коду ASCII

CINT (числ_выражение) округляет числовое выражение в пределах от -32768 до 32767 до целого значения.

CLNG (числ_выражение) округляет числовое выражение в пределах от -2147483648 до 2147483647.до длинного (4 байта) целого значения.

COS (угол). Возвращает косинус указанного в радианах угла.

CSNG (числ_выражение) переводит числовое выражение в значение с одинарной точностью.

CSRLIN. Возвращает текущую позицию строки курсора.

CVD (8_байт_числ_строка) переводит строку, образованную функцией MKD$, обратно в число двойной точности.

CVDMBF (8_байт_числ_строка) переводит строки, образованные функциями MKSMBF$ и MKDMBF$, обратно в число двойной точности в формате IEEE

CVI(2_байт_числ_строка), CVL (4_байт_числ_строка) и CVS (4_байт_числ_строка). Эти функции переводят строки, образованные функциями MKI$, MKL$ и MKS$, обратно в числа. CVI возвращает целое, CVL — длинное целое, CVS — число одинарной точности.

CVSMBF (4_байт_числ_строка) переводит строки, образованные функциями MKSMBF$ и MKDMBF$, обратно в число одинарной точности в формате IEEE

DATE$. Возвращает текущую системную дату компьютера в формате ММ-ДД-ГГГГ

**ENVIRON$**(переменная_среды$ | n%). Возвращает строку среды DOS

EOF(номер_файла%) проверка конца указанного открытого файла.

ERDEV. Возвращает код ошибки от последнего устройства, выдавшего ошибку. Младший байт возвращаемого значения, содержит код ошибки DOS. Старший байт содержит информацию об устройстве.

ERDEV$. Возвращает имя устройства, выдавшего эту ошибку.

ERL. Возвращает номер строки, в которой обнаружена ошибка, или последний установленный номер строки перед строкой, где была обнаружена ошибка. ERL не возвращает метку строки. Если нет номеров строк в программе, ERL возвращает 0

ERR. Возвращает код ошибки выполнения для последней ошибки.

EXP(числ_выражение). Возвращает е, возведенное в указанную степень, где е — основание натурального логарифма.Числ_выражение — число, меньшее или равное 88.02969

FILEATTR(номер_файла%,атрибут%). Возвращает информацию об открытом файле. Здесь номер_файла%" — номер открытого файла; атрибут%" — указывает тип возвращаемой информации. Если атрибут%" равен 1, FILEATTR возвращает значение, указывающее режим доступа к файлу (ввод, вывод, прямой, добавление, двоичный. Если атрибут%" равен 2(вывод), FILEATTR возвращает указатель файла DOS

FIX(числ_выражение) округляет выражение с плавающей запятой до его целой части.

FRE(числ_выражение | строк_выражение$). Возвращает объем (в байтах) доступной или неиспользуемой памяти. Здесь числ_выражение указывает тип памяти; строк_выражение$" любое строковое выражение. FRE сжимает свободное строковое пространство в один блок, а затем возвращает объем доступного строкового пространства.

FREEFILE. Возвращает номер следующего доступного неиспользуемого файла.

HEX$(числ_выражение&). Возвращает шестнадцатиричное представление числа. Числ_выражение&" любое числовое выражение. Выражение округляется до целого или длинного целого перед оценкой.

**INKEY$** считывает символ от клавиатуры.INKEY$ возвращает нулевую строку символов, если нет символа для возврата. Для стандартных клавиш INKEY$ возвращает 1-байтовую строку символов, содержащую считанный символ. Для расширенных клавиш INKEY$ возвращает 2-байтовую строку символов, состоящую из символа нуля (ASCII 0) и скан-кода клавиатуры.

INP(port%). Возвращает байт, считанный с аппаратного порта ввода/вывода. Port%" число в пределах от 0 до 65535, определяющее порт.

**INPUT$**(n[,[#]номер_файла%]). Возвращает строку символов, считанных из указанного файла. n число символов (байтов) для чтения; номер_файла%" номер открытого файла. Если этот параметр опущен, INPUT$ считывает с клавиатуры.

INSTR([начало%,]строк_выражение1$,строк_выражение2$). Возвращает позицию первого обнаружения строки в другой строке. Начало% устанавливает позицию символа, откуда начинается поиск. Если начало% опущено, INSTR начинает поиск с позиции 1; строк_выражение1$ строка, которую ищут; строк_выражение2$ строка, в которой ищут.

INT(числ_выражение). Возвращает наибольшее целое, меньшее или равное числовому выражению.

IOCTL [#]номер_файла%, строка$ посылает управляющую строку драйверу устройства. Здесь номер файла% — номер открытого устройства; строка$ — управляющая строка, посылаемая в устройство. Управляющая строка IOCTL зависит от драйвера устройства.

IOCTL$([#]номер_файла%). Возвращает информацию о текущем состоянии драйвера устройства.

LBOUND(массив[,размер%]). Возвращает верхнюю границу (наименьший доступ).

**LCASE$**(строк_выражение$) переводит все буквы строки в прописные буквы.

**LEFT$**(строк_выражение$,n%). Возвращает указанное число символов n%, стоящих с левого конца строки.

LEN(строк_выражение$ | переменная). Возвращает число символов в строке или число байтов, необходимых для хранения переменной.

LOC(номер_файла%). Возвращает текущую позицию в файле, где номер_файла% — номер открытого файла или устройства. Для бинарных файлов LOC возвращает позицию последнего считанного или записанного байта. Для файлов прямого доступа LOC возвращает номер последней записи, считанной из файла или записанной в файл. Для последовательных файлов LOC возвращает позицию текущего байта в этом файле, деленную на 128

LOF(номер_файла%). Возвращает длину файла в байтах.

LOG(числ_выражение). Возвращает натуральный логарифм числового выражения.

LPOS(n%). Возвращает число символов, посланных на печать после последнего символа возврата каретки. Здесь n% указывает один из портов принтера.

**LTRIM$**(строк_выражение$) удаляет начальные пробелы из строки.

**MID$**(строк_выражение$,начало%[,длина%]). Возвращает часть строки (подстроку) от символа в позиции начало% на заданную длину.

MKD$(выражение_двойной_точности#) переводит числа в 8-байтовую строку, которая может храниться в операторе строковых переменных FIELD. CVD переводит эту строку обратно в число.

MKDMBF$(выражение_двойной_точности#) переводит число в формате IEEE в числовую строку бинарного формата Microsoft (Microsoft-Binary-format), которая может храниться в операторе строковых переменных FIELD. CVDMBF переводит эту строку обратно в число в формате IEEE.

**MKI$**(целое_выражение%), MKL$(длинное_целое_выражение&), MKS$(выражение_одинарн_точности!) переводят числа в числовые строки, которые могут храниться в операторе строковых переменных FIELD. CVI, CVL, CVS переводят эти строки обратно в числа.

MKSMBF$(выражение_одинарн_точности!) переводит число в формате IEEE в числовую строку бинарного формата Microsoft (Microsoft-Binary-format), которая может храниться в операторе строковых переменных FIELD. CVSMBF переводит эту строку обратно в число в формате IEEE.

OCT$(числ_выражение&). Возвращает восьмеричное представление числа.

PEEK(адрес). Возвращает значение байта, хранящегося в указанном месте памяти.

PEN(n%). Возвращает статус светового пера. Перменная n% указывает информацию, которая должна быть возвращена о статусе светового пера.

PLAY(n). Возвращает число музыкальных нот в фоновой очереди, n — любое числовое выражение.

PMAP (начальная_координата#, n%). Возвращает координату окна эквивалентную координате области просмотра, определенную оператором WINDOW, или наоборот.

POINT {(n%) | (x%,y%)}. Возвращает текущие графические координаты курсора и атрибут цвета заданной точки растра (элемента растра). (n%) указывает тип возвращаемых координат, (x%,y%) — координаты точки растра, которую POINT проверяет на цвет. Если координаты находятся за пределами текущей области просмотра, POINT возвращает -1

POS(выражение). Возвращает текущую позицию столбца курсора.

**RIGHT$**(строк_выражение$,n%). Возвращает указанное (n%) число символов, стоящих с правого конца строки.

RND[(n#)]. Возвращает случайное число одинарной точности между 0 и 1, здесь n# — значение, устанавливающее то, как RND генерирует следующее случайное число.

**RTRIM$**(строк_выражение$). Удаляют конечные пробелы из строки.

SCREEN (строка%,столбец% [,флаг_цвета%]). Возвращает ASCII значение или атрибут цвета символа, находящегося в указанном месте экрана. Переменные имеют следующие значения: строка% — координата строки символа; столбец% — координата столбца символа; флаг_цвета% — значение (0 или 1), указывающее что возвращать.

SEEK(номер_файла%). Возвращает текущую позицию файла.

SGN(числовое_выражение). Возвращает значение, указывающее знак числового выражения (1 — если выражение положительно; 0 — если оно равно нулю; -1 — если оно отрицательно).

SIN(угол). Возвращает синус указанного в радианах угла.

SPACE$(n%). Возвращает строку пробелов.

SPC(n%) оставляет заданное число пробелов в операторах PRINT или LPRINT

SQR(числ_выражение). Возвращает квадратный корень числового выражения.

STICK(n%). Возвращает координаты джойстика, переменной n% задается возвращаемая коор-

дината (например, 0 — координата X джойстика A).

STR$(числ_выражение). Возвращает строковое представление числа.

STRIG(n%). Возвращает состояние триггера джойстика. Параметр указывает условие состояния джойстика (1 — нижний триггер джойстика A нажат в данный момент).

**STRING$**(длина%,{ascii-код% | строк_выражение$}). Возвращает строку указанной длины, состоящую из повторяющегося символа,заданного переменной ascii-код%, или первым символом из строк_выражения$.

TAB(столбец%) перемещает курсор текста в указанную позицию печати.

TAN(угол). Возвращает тангенс данного в радианах угла.

TIME$. Возвращает текущее системное время компьютера в формате чч:мм:с.

TIMER. Возвращает число секунд, прошедших с полуночи и до текущего момента. Используйте TIMER для хронометража программ или частей программ, или с оператором RANDOMIZE для запуска генератора случайных чисел.

UBOUND(массив[,размер%]). Возвращает нижнюю границу (наибольший доступный индекс) для указанного размера массива.

**UCASE$**(строк_выражение$). Переводит все буквы строки в строчные буквы.

VAL(строк_выражение$). Переводит строковое представление числа в число.

VARPTR(имя_переменной). Возвращает адрес смещения переменной.

**VARPTR$**(строка_команд$). Возвращает строковое представление адреса переменной для использования с операторами DRAW и PLAY. (строка_команд$) строковая переменная, содержащая команды DRAW или PLAY.

VARSEG(имя_переменной). Возвращает адрес сегмента переменной.

Операторы языка QBASIC

AND. Конъюнкция в булевых операциях.

BEEP. Генерирует звуковой сигнал через динамик компьютера.

BLOAD файл_спец$[,смещение%]. Загружает файл, созданный BSAVE, в память. Здесь файл_спец$ — файл образа памяти, созданный предыдущим BSAVE; смещение% — смещение начального адреса области памяти, в которую будут загружены данные.

BSAVE файл_спец$, смещение%, длина&. Копирует содержимое области памяти в файл. Здесь файл_спец$ — файл, в который байт за байтом будет скопирована область памяти; смещение% — смещение начального адреса области памяти, которую нужно сохранить; длина& — число байтов для копирования (от 0 до 65535). Начальный адрес области памяти для сохранения или загрузки определяется смещением начального адреса (offset) и последним оператором DEF SEG

[CALL] имя [([аргументы])]. Передает управление в процедуру типа SUB. Здесь имя — имя вызываемой процедуры типа SUB; аргументы — переменные или константы, передаваемые в процедуру SUB. Разделяйте аргументы запятыми. При указании аргументов массива введите имя массива и после него пустые скобки. Если опущено ключевое слово CALL, то пропустите также скобки, в которые заключен список аргументов, либо опишите процедуру в операторе DECLARE перед ее вызовом, либо сохраните программу и QBASIC автоматически сгенерирует оператор DECLARE. При указании аргумента, чье значение не будет изменяться процедурой, заключите этот аргумент в скобки.

CALL ABSOLUTE ([аргументы,] смещение%). Передает управление в процедуру на языке Ассемблер. Здесь аргументы — аргументы, переданные в процедуру на языке Ассемблер, такие, как смещения относительно текущего сегмента данных; смещение% — смещение от текущего кода сегмента, установленного DEF SEG, до выполнения процедуры.

CHAIN файл_спец$. Передает управление из текущей программы в другую программу BASIC. Здесь файл_спец$ — имя программы, в которую передается управление.

CHDIR путь$. Изменяет каталог устройства, принимаемый по умолчанию; путь$ — путь нового каталога, принимаемого по умолчанию, создаваемого подкаталога или удаляемого каталога.

CIRCLE [STEP] (x!,y!),радиус![,[цвет%] [,[старт!] [,[конец!] [,аспект!]]]]. Рисует на экране окружности и эллипсы; STEP — указывает, что координаты задаются по отношению к текущей графической позиции курсора; (x!,y!) — координаты центра окружности или эллипса; радиус! — радиус окружности или эллипса в единицах текущей координат, определенной последними операторами SCREEN, VIEW и WINDOW; цвет% — атрибут цвета, устанавливающий цвет окружности. Допустимые атрибуты цвета зависят от вашего графического адаптера и режима экрана, установленного последним оператором SCREEN; старт! — начальный угол дуги в радианах; конец! — конечный угол дуги в радианах; аспект — отношение длины оси У к длине оси X, используемое при изображении эллипсов. Для перевода градусов в радианы умножьте значение в градусах на (PI/180).

CLEAR [,,стек&]. Закрывает все файлы, освобождает буферы файлов, очищает все общие переменные, устанавливает числовые переменные и массивы в ноль, устанавливает строковые переменные в ноль и инициализирует стек. Кроме того, CLEAR может изменять размер стека. Переменная стек& устанавливает размер (в байтах) стекового пространства для ваших программ.

CLOSE [[#]номер_файла%[,[#]номер_файла%]...].Закрывает один или несколько файлов или устройств; номер_файла% — номер открытого файла или устройства. CLOSE без аргументов закрывает все открытые файлы и устройства.

CLS [{0 | 1 | 2}]. Очищает экран. CLS очищает либо текстовую, либо графическую области просмотра. Если была установлена графическая область просмотра (с помощью VIEW), то очищает только графическую область просмотра. В противном случае очищает текстовую область просмотра или весь экран. CLS 0 полностью очищает экран, как текст, так и графику. CLS 1 очищает графическую область просмотра или весь экран, если графическая область просмотра не была установлена. CLS 2 очищает текстовую область просмотра.

COLOR. Устанавливает цвета экрана. Может быть до трех параметров в зависимости от режима экрана.

COM(n%) ON | OFF | STOP. Включает, выключает или приостанавливает отслеживание событий в коммуникационном порте. n%-номер COM (серийного) порта (1 или 2).

COMMON [SHARED] переменные. Определяет глобальные переменные, которые могут быть использованы совместно несколькими программами или в цепочке программ. SHARED обозначает, что переменные совместно используются всеми процедурами SUB и FUNCTION.

CONST имя_константы = выражение [,имя_константы = выражение]... Описывает одну или несколько символьных констант. Выражение может состоять из буквальных констант (например, 1.0), других констант, любых арифметических и логических знаков операций, исключая возведение в степень (^), или одной строки литералов.

DATA константа[,константа]... Указывает значения для чтения последующими операторами READ. Операторы DATA могут быть введены только на модульном уровне. Они не могут использоваться в процедурах.

DATE$ мм-дд-гг | мм-дд-гггг | мм/дд/гг | мм/дд/гггг. Устанавливает текущую системную дату в вашем компьютере в указанном формате.

DECLARE {FUNCTION | SUB} имя [([параметры])]. Описывает процедуру FUNCTION или SUB и выполняет проверку аргументов на соответствие типу данных. DECLARE необходим при вызове процедур SUB без использования CALL. QBASIC автоматически добавляет оператор DECLARE при сохранении программы.

DEF FNимя[(параметры)] = выражение функции. Определяет функцию. (параметры) — один или несколько аргументов.

DEF SEG [=адрес]. Устанавливает адрес текущего сегмента. Здесь адрес — адрес сегмента, используемый BLOAD, BSAVE, CALL ABSOLUTE, PEEK или POKE; значение в пределах от 0 до 65535. Если адрес опущен, DEF SEG переустанавливает адрес сегмента в сегмент данных, принимаемый по умолчанию.

DEFINT обл_букв [,обл_букв]...; DEFLNG обл_букв [,обл_букв]...; DEFSNG обл_букв [,обл_букв]...; DEFDBL обл_букв [,обл_букв]...; DEFSTR обл_букв [,обл_букв]... Устанавливают тип данных по умолчанию для переменных, функций DEF FN и процедур FUNCTION. Здесь обл_букв — буква или область букв (например, A-M). QBASIC устанавливает тип данных по умолчанию для переменных, функций DEF FN и процедур FUNCTION, имена которых начинаются с указанной буквы, в следующем порядке: DEFINT целый; DEFLNG длинное; DEFSNG одинарной точности; DEFDBL двойной точности; DEFSTR строковый. Однако суффикс типа данных (%, &, !, # или $) всегда имеет преимущество над операторами DEFтип. Одинарная точность — тип данных по умолчанию, если вы не указали операторов DEFтип. После указания в программе оператора DEFтип QBASIC автоматически вставляет соответствующий оператор DEFтип в каждую создаваемую процедуру.

DIM [SHARED] переменная[(индекс)] [AS тип] [,переменная[(индекс)] [AS тип]]... Описывает массив или указывает тип данных для переменных не массива. SHARED указывает, что переменные используются совместно всеми процедурами SUB или FUNCTION в модуле; переменная — имя переменной массива; индекс — размеры массива в следующем виде: [низ TO] верх [,[низ TO] верх]..., где низ — нижняя граница индекса массива. По умолчанию нижняя граница равна нулю; верх — верхняя граница. AS тип — описывает тип данных массива или переменной (INTEGER, LONG, SINGLE, DOUBLE, STRING или тип данных, описанный пользователем). DIM описывает либо статические, либо динамические массивы. Если область хранения массива не была описана в $STATIC, $DYNAMIC или COMMON, массивы, размер которых был задан в числах, являются статическими, а массивы, размер которых был задан в переменных, — динамическими. REDIM всегда описывает динамические массивы. Область хранения статического массива выделяется при запуске программы и остается постоянной. Область хранения динамического массива выделяется при работе программы.

DO [{WHILE | UNTIL} условие] [блок_операторов] LOOP. Повторяет блок операторов, пока условие верно, или до тех пор, пока условие не станет верным. Условие может быть задано как после DO, так и после LOOP.

DRAW строка_команд$. Рисует объект. Здесь строка_команд$ — строковое выражение, содержащее одну или несколько следующих команд DRAW.

Команды изображения линии и перемещения курсора:

D[n%] — перемещает курсор вниз на n% единиц;

E[n%] — перемещает курсор вправо-вверх на n% единиц;

F[n%] — перемещает курсор вправо-вниз на n% единиц;

G[n%] — перемещает курсор влево-вниз на n% единиц;

H[n%] — перемещает курсор влево-вверх на n% единиц;

L[n%] — перемещает курсор влево на n% единиц;

M{+|-}]x%,y% перемещает курсор в точку x%,y%. Если перед x стоит + или -, перемещает относительно текущей точки;

R[n%] — перемещает курсор вправо на n% единиц;

U[n%] — перемещает курсор вверх на n% единиц;

[B] — необязательная приставка, которая перемещает курсор, не рисуя при этом;

[N] — необязательная приставка, которая рисует и возвращает курсор в первоначальную позицию.

Команды цвета, вращения и масштаба:

An% — поворачивает объект на n% * 90 градусов (n% быть 0, 1, 2 или 3);

Cn% — устанавливает рисующий цвет (n% — атрибут цвета);

Pn1%, n2% — устанавливает цвет заполнения и границы объекта (n1% — атрибут цвета заполнения, n2% — атрибут цвета границы);

Sn% — определяет масштаб рисунка, устанавливая единицу длины перемещения курсора, по умолчанию n% равно 4, что эквивалентно 1-й точке растра;

TAn% — поворачивает угол на n% градусов (от -360 до 360).

Если вы опустили n% в командах изображения линии и перемещения курсора, курсор перемещается на 1 единицу. Для выполнения подстроки команд DRAW из строки команд DRAW используйте команду «X»: DRAW «X»+ VARPTR$(строка_команд$)

END [{DEF | FUNCTION | IF | SELECT | SUB | TYPE}]. Заканчивает при: DEF состоящее из нескольких строк, определение функции DEF FN; FUNCTION определение процедуры типа FUNCTION; IF блок оператора IF...THEN...ELSE; SELECT блок SELECT CASE; SUB процедуру типа SUB; TYPE определение типа данных, определенных пользователем. Если аргумент не указан, то

END завершает программу и закрывает все открытые файлы.

ENVIRON строк_выражение$. Изменяет или добавляет строку среды в таблицу среды DOS. Здесь строк_выражение$ — имя или установка переменной среды DOS (например, PATH или PROMPT) в одном из следующих форматов: переменная_среды$ = строка_среды$; переменная_среды$ — строка_среды$. Изменения, сделанные оператором ENVIRON, удаляются по окончании работы QBASIC.

EQV. Эквивалентность в булевых операциях.

ERASE. Имя_массива [,имя_массива]... Переинициализирует элементы массива или освобождает область хранения динамического массива. В статических массивах ERASE обнуляет каждый элемент числового массива, а каждый элемент строкового массива превращает в строку нулевой длины. Для динамических массивов ERASE освобождает память, используемую массивом. Вам придется заново описать размеры массива в REDIM или DIM, прежде чем использовать его.

ERROR выражение%. Эмулирует ошибку BASIC или ошибку, определенную пользователем. Здесь выражение% — код ошибки BASIC или ошибки, определенной пользователем; значение в интервале от 1 до 255. Для определения собственного кода ошибки используйте значения, которых нет в таблице.

EXIT {DEF | DO | FOR | FUNCTION | SUB}. Выход из цикла DO или FOR, процедуры типа FUNCTION или SUB, или функции типа DEF FN.

FIELD [#]номер_файла%, ширина_поля% AS переменная$ [,ширина_поля% AS переменная$] ... Отводит место под переменные в буфере файлов прямого доступа. Здесь номер_файла% — номер открытого файла; ширина_поля% число символов в поле; переменная$ переменная, идентифицирующая поле и содержащая данные поля. Запись переменных обычно обеспечивает лучший способ для управления записью данных.

FILES [файл_спец$]. Отображает содержимое текущего каталога или указанного каталога. Здесь файл_спец$ — имя файла или путь (может включать устройство и подстановочные символы DOS). Если вы не указываете файл_спец$, то FILES отображает все файлы в текущем каталоге.

FOR счетчик = начало TO конец [STEP приращение]
    [блок_операторов]
NEXT [счетчик [,счетчик]...]. Повторяет блок операторов указанное число раз. Здесь счетчик — числовая переменная, используемая как счетчик цикла; начало и конец — начальное и конечное значение счетчика; приращение — приращение счетчика при каждом шаге цикла.

FUNCTION. Определяет процедуру FUNCTION. Процедура определяется следующим образом:
FUNCTION имя [(параметры)] [STATIC]
    [блок_операторов]
  имя = выражение
    [блок_операторов]
END FUNCTION. Здесь имя — имя функции и возвращаемый ею тип данных, указанный суффиксом типа данных (%, &, !, # или $); параметры — одна или несколько переменных, указывающих параметры, которые передаются в функцию при ее вызове; STATIC указывает, что значения локальных переменных функции сохраняются между вызовами функции; выражение — возвращаемое значение функции. При вызове функции можно указать, что значение аргумента не должно изменяться функцией. Для этого заключите аргумент в круглые скобки.

GET [#]номер_файла%[,[номер_записи&] [,переменная]]. Считывает из файла в буфер прямого доступа или в переменную. Здесь номер_файла% — номер открытого файла; номер_записи& — для файлов прямого доступа — номер записи для чтения. Для двоичных файлов — позиция байта, где начинается чтение; переменная — переменная, используемая для приема ввода из файла. Переменной обычно является переменная определенного пользователем типа данных.

GET [STEP](x1!,y1!)-[STEP](x2!,y2!), имя_массива[(индекс%)]. Сохраняет графический образ экрана. STEP указывает, что координаты задаются относительно текущей графической позиции курсора; (x1!,y1!) — координаты левого верхнего угла образа, сохраненного GET; (x2!,y2!) — координаты правого нижнего угла сохраненного образа; имя_массива — имя массива, в котором хранится образ для определения требуемого размера массива; индекс% — индекс массива, где начинается хранение образа.

GOSUB строка1... RETURN [строка2]. Передает управление в подпрограмму и возвращается из нее. Здесь строка1 — метка или номер первой строки подпрограммы; строка2 — метка или номер строки, в которую возвращается подпрограмма. Если вы не указываете метку или номер строки для RETURN, то выполнение программы продолжится с оператора, стоящего за GOSUB (при вызове подпрограммы), или там, где произошло событие (при обработке событий). Смотрите ключевое слово ON для информации об операторах обработки событий. Операторы SUB и CALL представляют собой более удобную альтернативу для подпрограмм GOSUB.

GOTO строка. Выполняет переход в указанную строку. DO...LOOP, SELECT CASE, IF...THEN...ELSE, SUB и FUNCTION предоставляют более удобный способ контроля хода вашей программы. GOTO также используется в качестве ключевого слова в операторе ON ERROR.

IF... THEN... ELSE. Выполняет оператор или блок операторов в зависимости от указанных условий и имеет следующую структуру:
IF условие1 THEN
    [блок_операторов 1]
[ELSEIF условие2 THEN
    [блок_ операторов 2]]...
[ELSE
    [блок операторов n]]
END IF. Здесь условие1 и условие2 — любые выражения, которые могут быть оценены как истинное (не-ноль) или ложное (ноль); блок_операторов 1, блок_операторов 2 и блок_операторов n — один или несколько операторов в одной или нескольких строках; операторы — один или несколько операторов, разделенных двоеточиями.

IMP. Импликация в булевых операциях.

INPUT [;] ["приглашение"{; | ,}] список переменных. Считывает входные данные от клавиатуры. Здесь приглашение — необязательная строка, отображаемая на экране перед вводом пользователем данных. Точка с запятой, стоящая сразу после приглашения, добавляет знак вопроса в строку приглашения; список переменных — одна или несколько переменных, разделенных запятыми, в которых хранятся данные, введенные от клавиатуры. Переменная может состоять из 40 символов и должна начинаться с буквы. Допустимыми являются символы A-Z, 0-9 и точка (.); переменная$ —

хранит строку символов, введенных от клавиатуры. При вводе от клавиатуры точка с запятой, стоящая сразу после INPUT, оставляет курсор в той же строке по нажатии пользователем клавиши Enter. INPUT использует запятую для разделения записей.

INPUT #номер файла%, список переменных. Считывает входные данные из файла, номер файла% — номер открытого файла.

IOCTL [#]номер_файла%, строка$. Посылает управляющую строку драйверу устройства. Здесь номер файла% — номер открытого устройства; строка$ — управляющая строка, посылаемая в устройство. Управляющая строка IOCTL зависит от драйвера устройства.

KEY {сл_слово | key%, символьное_выражение$}. Присваивает клавише строку символов и может отображать значения клавиш. Здесь сл_слово — или LIST (отображает установки для каждой клавиши), или ON (включает строку отображения функциональных клавиш), или OFF (выключает строку отображения функциональных клавиш); клавиша% — номер функциональной клавиши. Используйте цифры от 1 до 10 для функциональных клавиш от F1 до F10. Используйте 30 и 31 для функциональных клавиш F11 и F12 на расширенной клавиатуре; символьное_выражение$ — строка длинной до 15 символов, которая отображается при нажатии функциональной клавиши.

KEY(n%) сл_слово. Включает, выключает или приостанавливает отслеживание событий для клавиш. Здесь n% — значение, указывающее на функциональную клавишу, клавишу направления или определенную пользователем клавишу; сл_слово — ON | OFF | STOP | GOSUB строка, соответственно включает отслеживание событий для указанной клавиши, OFF выключает отслеживание событий для клавиш, STOP приостанавливает отслеживание событий, GOSUB строка переводит на первую строку подпрограммы отслеживания событий.

KILL файлы$. Удаляет файлы с диска. Здесь файлы$ — определяет файл или файлы для удаления. Может включать путь и подстановочные символы DOS (? и *).

[LET] переменная=выражение. Присваивает значение выражения переменной.

LINE [[STEP](x1!,y1!)]-[STEP](x2!,y2!) [,[цвет%] [,[B | BF] [,стиль%]]]. Рисует на экране линию или прямоугольник. STEP указывает, что координаты задаются относительно текущего графического положения курсора; (x1!,y1!) и (x2!,y2!) — координаты начала и конца линии на экране; цвет% — атрибут цвета, устанавливающий цвет линии или прямоугольника. Допустимые атрибуты цвета зависят от вашего графического адаптера и режима экрана, установленного последним оператором SCREEN. B — рисует прямоугольник вместо линии; BF — рисует заполненный прямоугольник; стиль% — 16-битовое значение, биты которого устанавливают, будут ли рисоваться точки растра. Используется для изображения прерывистых и пунктирных линий.

LINE INPUT [;] ["приглашение";] переменная$. Считывает строку, содержащую до 255 символов, от клавиатуры. Здесь приглашение необязательная строка, отображаемая на экране перед вводом пользователем данных. Точка с запятой, стоящая сразу после приглашения, добавляет знак вопроса в строку приглашения; список переменных — одна или несколько переменных, разделенных запятыми, в которых хранятся данные, введенные от клавиатуры; переменная$ — хранит строку символов, введенных от клавиатуры. LINE INPUT считывает все символы до символа возврата каретки.

LINE INPUT #номер файла%, переменная$. Считывает строку, содержащую до 255 символов, из файла. Здесь номер файла% — номер открытого файла; переменная$ — хранит строку символов, считанных из файла.

LOCATE [row%] [,[column%] [,[curcor%] [,start% [,stop%]]]]. Перемещает курсор на экране в указанную позицию. Здесь row% и column% — номер строки и столбца, куда перемещается курсор; cursor% — определяет режим отображения курсора: 0 = видимый, 1 = невидимый; start% и stop% — целое выражение в пределах от 0 до 31, которое определяет первую и последнюю строки сканирования курсора. Вы можете изменить размер курсора посредством изменения строк сканирования курсора.

LOCK [#]номер_файла% [,{запись& | [начало&] TO конец&}]. Ограничивает или закрывает доступ к файлу при работе в сети. Здесь номер_файла% — номер открытого файла; запись& — для файлов прямого доступа — номер закрываемой записи по отношению к первой записи в файле; для бинарных файлов — номер закрываемого байта относительно первого байта в файле; начало& и конец& — номера первой и последней записи или байта в диапазоне закрываемых записей или байтов. Оператор UNLOCK отменяет ограничения, наложенные последним оператором LOCK. Для последовательных файлов LOCK и UNLOCK влияют на весь файл.

LPRINT [выражения] [{; | ,}]. Печатает данные на принтере LPT1. Здесь выражения — одно или несколько числовых или символьных выражений для печати; {; | ,} — определяет, где начинается следующий вывод: ; — печать сразу после последнего значения; , — печать в начале следующей зоны печати. Ширина зоны печати — 14 символов.

LPRINT USING символы_формата$; выражения [{; | ,}]. Печатает отформатированный вывод на принтере LPT1. Здесь номер_файла% — номер следующего открытого файла; символы_формата$; — символьное выражение, содержащее один и более «определителей формата»; выражения — одно или несколько числовых или символьных выражений, разделенных запятыми, точкой с запятой, пробелами или табуляцией; {; | ,} — определяет, где начинается следующий вывод: ; — печать сразу после последнего значения; , — печать в начале следующей зоны печати. Ширина зоны печати — 14 символов.

LSET строк_переменная$ = строк_выражение$ | запись_переменной1 = запись_переменной2. Перемещает данные в буфер файлов прямого доступа (подготавливая его к оператору PUT) и осуществляет левое выравнивание значений строковой переменной, а также копирует содержимое одной записи переменной в другую. Здесь строк_переменная$ — любая строковая переменная или поле файлов прямого доступа, определенные в операторе FIELD; строк_выражение$ — версия строк_переменной$ для левого края; запись_переменной1 и запись_переменной2 — запись переменных любого определенного пользователем типа данных. Используйте LSET для присвоения записи переменной одного типа данных другому типу данных, определенному пользователем.

**M!D$**(строк_переменная$,начало%[,длина%]) =строк_выражение$. Замещает часть строковой переменной другой строкой. Здесь строк_переменная$ — изменяемая строковая переменная; начало% — позиция первого возвращаемого или

замещаемого символа в строке; длина% — число символов в подстроке. Если длина опущена, MID$ возвращает или замещает все символы справа от начальной позиции; строк_выражение$ — строка замещения, может быть любым строковым выражением.

MKDIR путь$. Создает подкаталог.

MOD. Делит одно число на другое и возвращает остаток. Синтаксис: числ_выражение1 MOD числ_выражение2, где числ_выражение1 и числ_выражение2 — любые числовые выражения. Действительные числа округляются до целых.

NAME старый$ AS новый$. Переименовывает файлы и каталоги. Каждое имя файла может включать путь.

NOT. Отрицание в булевых операциях.

ON COM(n%) GOSUB строка. При включенном отслеживании событий обращается к подпрограмме каждый раз, как только порт получает символы.

ON ERROR GOTO строка. Включает обработку ошибок при обнаружении ошибки во время выполнения программы, передавая управление на подпрограмму обработки ошибок, или возобновляет выполнение программы. Для выключения обработки ошибок задайте: GOTO 0. Если оператор ON ERROR не используется, любая ошибка выполнения завершает вашу программу. Здесь n% — значение, указывающее на функциональную клавишу, клавишу направления или определенную пользователем клавишу.

ON PEN GOSUB строка. При включенном отслеживании события ON PEN обращается к подпрограмме при каждой активизации светового пера.

ON PLAY(лимит_очереди%) GOSUB строка. При включенном отслеживании события ON PLAY обращается к подпрограмме каждый раз, когда в музыкальном буфере содержится число нот, меньшее указанного числа. Здесь лимит_очереди% — число в пределах от 1 до 32. ON PLAY обращается к подпрограмме, когда число нот в музыкальном буфере меньше числа лимит_очереди%; строка метка или номер первой строки подпрограммы отслеживания события.

ON STRIG(n%) GOSUB строка. При включенном отслеживании события ON STRIG обращается к подпрограмме при каждом нажатии указанного триггера. Здесь n% — значение, указывающее триггер джойстика.

ON TIMER(n%) GOSUB строка. При включенном отслеживании события ON TIMER обращается к подпрограмме, как только прошло указанное число секунд. Здесь n% — число секунд, которые должны пройти, прежде чем ON TIMER обратится к подпрограмме отслеживания события; значения находятся в пределах от 1 до 86400 (24 часа).

ON выражение% GOSUB список_строк и ON выражение% GOTO список_строк. Выполняют переход в одно из нескольких положений в зависимости от значения выражения. Здесь выражение% — выражение в диапазоне от 0 до 255; список_строк — набор меток или номеров строк. Если значение выражения равно 1, то программа переходит в первую строку из списка; если выражение равно 2, то программа переходит во вторую строку из списка и т.д. SELECT CASE предоставляет более удобный путь для ветвления программы.

OPEN файл$ [FOR режим] [ACCESS доступ] [закр] AS [#]номер_файла% [LEN=д_зап%]. Открывает файл или устройство. Здесь файл$ — имя файла или устройства. Имя файла может включать устройство и путь; режим — один из следующих режимов файла: APPEND, BINARY, INPUT, OUTPUT или RANDOM; доступ — при работе в сети указывает, открыт ли файл для READ (чтения), WRITE (записи) или READ WRITE (чтения-записи); закр — указывает, как файл закрыт для сетевого доступа: SHARED (общий), LOCK READ (закрыт для чтения), LOCK WRITE (закрыт для записи), LOCK READ WRITE (закрыт для чтения-записи); номер_файла% — номер в пределах от 1 до 255, идентифицирующий открытый файл; д_зап% — для файлов прямого доступа — длина записи (по умолчанию 128 байт). Для последовательных файлов — число буферированных символов (по умолчанию 512 байт).

OPEN «COMn: параметры1 параметры2" [FOR режим] AS [#]filenum% [LEN=reclen%]. Открывает и инициализирует канал связи для ввода или вывода (В/ВВ). Оператор OPEN COM должен быть выполнен до начала работы с устройством, использующим для связи интерфейс RS232. Здесь n — открываемый порт связи (1 = COM1, 2 = COM2); параметры1 — наиболее часто используемые параметры связи: [бод] [,[паритет] [,[данные] [,[стоп]]]] ; бод — скорость передачи данных (в бодах) открываемого устройства: 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600; паритет — вариант контроля паритета: N (отсутствует), E (по четности), O (по нечетности), S (с промежутком), M (с маркером), PE (включение проверки ошибок); данные — число битов данных в байте: 5, 6, 7, 8; стоп — число стоповых битов: 1, 1.5, 2. По умолчанию: 300 бод, по четности, 7 битов данных, 1 стоповый бит; параметры2 — список реже используемых параметров, разделяемых запятыми: ASC открывает устройство в режиме ASCII, BIN открывает устройство в двоичном режиме, CD[m] устанавливает тайм-аут (в миллисекундах) для линии DCD (Data Carrier Detect), CS[m] устанавливает тайм-аут (в миллисекундах) для линии CTS (Clear to Send), DS[m] устанавливает тайм-аут (в миллисекундах) для линии DS (Data Set Ready), LF посылает символ перевода строки после символа возврата каретки, OP[m] задает время ожидания (в миллисекундах) OPEN COM открытия каналов связи, RB[n] устанавливает размер (в байтах) принимающего буфера, RS подавляет обнаружение сигнала »Запрос на передачу" (RTS), TB[n] устанавливает размер (в байтах) передающего буфера; режим INPUT, OUTPUT, или RANDOM (по умолчанию); filenum% — число в пределах от 1 до 255, идентифицирующее канал связи, пока он открыт; reclen% — размер буфера режима прямого доступа (по умолчанию 128 байтов).

OPTION BASE {0 | 1}. Устанавливает нижнюю границу индекса массива, принимаемую по умолчанию.

OR. Дизъюнкция (включающее «или») в булевых операциях.

OUT port%, data% посылает байт в аппаратный порт ввода/вывода. Здесь port% — число в пределах от 0 до 65535, определяющее порт; data% — числовое выражение в пределах от 0 до 255, посылаемое в порт.

PAINT [STEP] (x!,y!)[,{цвет% | образец$}] [,[цвет_границ%] [,фон$]]]. Заполняет графическую область указанным цветом или образцом. Здесь STEP указывает, что координаты задаются относительно текущей графической позиции курсора; (x!,y!) — координаты экрана начала зарисовки; color% — атрибут цвета, устанавливающий заполняющий цвет; образец$ — образец заполнения

шириной 8 бит и длиной до 64 байт, определяемый следующим образом: образец$ = CHR$(arg1) + CHR$(arg2) + ... + CHR$(argn%). Аргументы CHR$ — числа в пределах от 0 до 255. Каждый CHR$(argn%) определяет 1 — байт, участок в 8 пикселов образца, основанного на бинарной форме числа; цвет_границ% — атрибут цвета, указывающий цвет границы заполняемой области. PAINT прекращает заполнение области, когда он наталкивается на границу указанного цвета; фон$ — 1 байт, фон участка в 8 пикселов образца. Указание фона участка образца позволяет вам зарисовывать уже зарисованную область. Допустимые атрибуты цвета зависят от вашего графического адаптера и режима экрана, установленного последним оператором SCREEN.

PALETTE [атрибут%,цвет&]

PALETTE USING имя_массива#[(индекс%)]. Изменяют установки атрибутов цвета в текущем режиме экрана. PALETTE и PALETTE USING работают только в системах с адаптерами EGA, VGA или MCGA. Здесь атрибут% — изменяемый атрибут цвета; цвет& — значение цвета, присвоенное атрибуту; имя_массива# — массив значений цвета, который присваивается текущей группе атрибутов цвета экрана. Массив должен быть достаточно большим для присвоения цветов всем атрибутам; индекс% индекс первого элемента массива, который присваивается атрибуту. Допустимые атрибуты и значения цвета зависят от Вашего графического адаптера и режима экрана, установленного последним оператором SCREEN.

PCOPY исх_страница%,приним_страница%. Копирует одну страницу видеопамяти в другую. Здесь исх_страница% — номер копируемой страницы видеопамяти; приним_страница% — номер страницы видеопамяти, в которую копируют. Значение, идентифицирующее видеостраницу, определяется размером видеопамяти и текущим режимом экрана.

PEN ON | OFF | STOP. Включает, выключает или приостанавливает отслеживание события для светового пера. При включенном отслеживании события ON PEN обращается к подпрограмме при каждой активизации светового пера. ON — включает отслеживание события для светового пера. OFF — выключает отслеживание события для светового пера. STOP — приостанавливает отслеживание события для светового пера. События обрабатываются при включении PEN ON отслеживания события.

PLAY строка_команд$. Звуковое воспроизведение по нотам. Здесь строка_команд$ — выражение, состоящее из одной или нескольких следующих команд PLAY.

Команды октавы и тона:

Октава — задает текущую октаву (0 — 6).

< или > — переходит на одну октаву вверх или вниз.

A — G — играет определенную ноту текущей октавы.

Nнота Играет определенную ноту (0 — 84) в диапазоне семи октав (0 — сброс).

Команды длительности и темпа:

Lразмер — задает длительность каждой ноты (1 — 64): L1 — целая нота, L2 — 1/2 ноты и т.д.

ML — вид исполнения legato.

MN — вид исполнения normal.

MS — вид исполнения staccato.

Pпауза — задает паузу (1 — 64). P1 — пауза в целую ноту, P2 — пауза в 12 ноты и т.д.

Tтемп — задает темп исполнения в четвертях в минуту (32-255).

Команды режима:

MF — основное звучание.

MB — фоновое звучание.

Команды изменения ноты:

или + — диез.

- (минус) — бемоль.

. (точка) — длительность 3/2 от размера ноты.

Для выполнения подстрок команды PLAY из строки команды PLAY используйте команду «X»: PLAY «X»+ VARPTR$(строка_команд$).

PLAY ON | OFF | STOP. Включает, выключает или приостанавливает отслеживание события при игре музыки.

PEEK. Возвращает значение байта, хранящегося в указанном месте памяти.

POKE адрес,байт%. Записывает значение байта в указанное место памяти. Здесь адрес — позиция байта относительно адреса текущего сегмента, установленного DEF SEG; значения в пределах от 0 до 65535; байт% — значение байта для записи в указанное место памяти; значения в пределах от 0 до 255.

PRESET [STEP] (x!,y!) [,color%]. Рисует заданную точку на экране. STEP — указывает, что x! и y! заданы относительно текущего графического положения курсора; (x!,y!) — координаты экрана устанавливаемой точки растра; color% — атрибут цвета, устанавливаемый для точки растра. Если color% опущен, PRESET использует текущий цвет фона. Доступные атрибуты цвета зависят от вашего графического видеоадаптера и режима экрана. Значения координат зависят от графического видеоадаптера, режима экрана и последних установок в операторах VIEW и WINDOW.

PRINT [#номер_файла%,] [выражения] [{; | ,}]. Выводит данные на экран или записывает их в файл. Здесь номер_файла% — номер открытого файла. Если номер файла не указан, PRINT выводит данные на экран; выражения — одно или несколько числовых или символьных выражений для печати; {; | ,} — определяет, где начинается следующий вывод: ; — печать сразу после последнего значения.; , — печать в начале следующей зоны печати. Ширина зоны печати — 14 символов.

PRINT [#номер_файла%,] USING символы_формата$; выражения [{; | ,}]. Записывает отформатированный вывод на экран или в файл. Здесь номер_файла% — номер следующего открытого файла; символы_формата$; — символьное выражение, содержащее один и более «определителей формата»; выражения — одно или несколько числовых или символьных выражений, разделенных запятыми, точкой с запятой, пробелами или табуляцией; {; | ,} — определяет, где начинается следующий вывод: ; — печать сразу после последнего значения; , — печать в начале следующей зоны печати. Ширина зоны печати — 14 символов.

PSET [STEP] (x!,y!) [,color%]. Рисует заданную точку на экране. STEP — указывает, что x! и y! заданы относительно текущего графического положения курсора; (x!,y!) — координаты экрана устанавливаемой точки растра; color% — атрибут цвета, устанавливаемый для точки растра. Если color% опущен, PSET использует текущий цвет переднего плана. Доступные атрибуты цвета зависят от Вашего графического видеоадаптера и режима экрана. Значения координат зависят от графического видеоадаптера, режима экрана и последних установок в операторах VIEW и WINDOW.

PUT [STEP] (x1!,y1!), имя_массива[(index%)] [,ключ_слово]. Выводит на экран образ, сохранен-

ный GET. STEP — указывает, что координаты задаются относительно текущей графической позиции курсора; (x1!,y1!) — положение на экране, где PUT выводит образ; имя_массива — имя массива, в котором хранится образ; индекс% — индекс массива, где начинается хранение образа; ключ_слово — ключевое слово, обозначающее, как отображается образ: AND — объединяет сохраненный образ с существующим образом, OR — накладывает сохраненный образ на существующий образ, PSET — рисует сохраненный образ, стирая существующий образ, PRESET — рисует сохраненный образ в реверсивном цвете, стирая существующий образ, XOR — рисует сохраненный образ или стирает ранее нарисованный образ, храня при этом фоновые эффекты мультипликации. Оператор PUT всегда должен выполняться в том же режиме экрана, в котором использовался оператор GET для сохранения образа или в совместимом режиме.

RANDOMIZE [число%]. Инициализирует генератор случайных чисел. Здесь число% — число, используемое для инициализации генератора случайных чисел. Если опущено, то RANDOMIZE запрашивает его.

READ переменные. Считывает значения из оператора DATA и присваивает их переменным, перечисленным через запятую.

REDIM [SHARED] переменная(индекс) [AS тип] ... [,переменная(индекс) [AS тип]]... Описывает или изменяет размер динамического массива, стирая все предыдущие значения. SHARED. Указывает, что переменные используются совместно всеми процедурами SUB или FUNCTION в модуле; переменная — имя переменной массива; индекс — размеры массива в следующем виде: [низ TO] верх [,[низ TO] верх]..., где низ — нижняя граница индекса массива. По умолчанию нижняя граница равна нулю, верх — верхняя граница. AS тип. Описывает тип данных массива или переменной (INTEGER, LONG, SINGLE, DOUBLE, STRING или тип данных, описанный пользователем). DIM описывает либо статические, либо динамические массивы. Если область хранения массива не была описана в $STATIC, $DYNAMIC или COMMON, массивы, размер которых был задан в числах, являются статическими, а массивы, размер которых был задан в переменных — динамическими. REDIM всегда описывает динамические массивы. Область хранения статического массива выделяется при запуске программы и остается постоянной. Область хранения динамического массива выделяется при работе программы.

REM комментарий — позволяет вставлять комментарии в программы. При работе программы комментарии игнорируются, если только они не содержат метакоманды. Комментарий может быть вставлен в строке после выполняемого оператора, если перед комментарием стоит символ апострофа (') или перед REM стоит двоеточие (:).

RESET. Закрывает все открытые файлы и устройства.

RESTORE [строка]. Позволяет READ заново считать значения в указанном операторе DATA. Здесь строка — метка или номер строки оператора DATA. Если строка опущена, следующий оператор READ считывает значения из оператора DATA, стоящего в программе первым.

RESUME [{строка | NEXT}]. Продолжает выполнение программы после подпрограммы обработки ошибок. Здесь строка — метка или номер строки для продолжения программы. Если метка равна 0 или отсутствует, выполнение продолжается с оператора, в котором была обнаружена ошибка; NEXT — продолжает выполнение программы со следующего оператора за оператором, в котором была обнаружена ошибка.

RETURN [строка2]. Используется в конструкции GOSUB... RETURN для возврата из подпрограммы. Здесь строка2 — метка или номер строки, в которую возвращается подпрограмма. Если Вы не указываете метку или номер строки для RETURN, то выполнение программы продолжится с оператора, стоящего за GOSUB (при вызове подпрограммы) или там, где произошло событие (при обработке событий).

RMDIR путь$. Удаляет подкаталог.

RSET строк_переменная$=строк_выражение $ — перемещают данные в буфер файлов прямого доступа (подготавливая его к оператору PUT) и осуществляют правое выравнивание значений строковой переменной. Здесь строк_переменная $ — любая строковая переменная или поле файлов прямого доступа, определенные в операторе FIELD; строк_выражение$ — версия строк_переменной$ для правого края; запись_переменной1 и запись_переменной2 — запись переменных любого определенного пользователем типа данных.

RUN [{номер_строки | файл$}]. Выполняет текущую или указанную программу. Здесь номер_строки — номер строки в текущей программе, где должно начинаться выполнение программы. Если номер строки не указан, то выполнение программы начнется с первой выполняемой строки; файл$ — имя исходного файла BASIC. QBASIC ожидает расширение .BAS. Перед загрузкой программы RUN закрывает все файлы и очищает память. Используйте оператор CHAIN для запуска программы, не закрывая при этом открытых файлов.

SCREEN режим% [,[перекл_цвета%] [,[актив_стр%] [,видим_стр%]]]. Устанавливает режим и другие характеристики экрана. Здесь режим% — устанавливает режим экрана; перекл_цвета% — значение (0 или 1), которое переключает между цветным и монохромным изображением (только в режимах 0 и 1); актив_стр% — страница экрана, в которую записывается вывод текста или графики; видим_стр% — страница экрана, отображаемая на экране в данный момент.

SEEK [#]номер_файла%, позиция&. Устанавливает позицию файла для следующей записи или чтения. Здесь номер_файла% — номер открытого файла; позиция — позиция, в которой будет осуществляться следующее чтение или запись. Для файлов прямого доступа, номер записи.Для других файлов, позиция байта по отношению к началу файла. Первый байт находится в позиции 1.

SELECT CASE. Выполняет один из нескольких блоков операторов в зависимости от значения выражения. Синтаксис:

```
SELECT CASE тест_выражение
CASE список_выражений1
  [блок_операторов-1]
[CASE список_выражений2
  [блок_операторов-2]]...
[CASE ELSE
  [блок_операторов-n]]
```

END SELECT. Здесь тест_выражение — любое числовое или строковое выражение; список_выражений1 и список_выражений2 — одно или несколько выражений для сравнения с тест_выражением. В выражении ключевое слово IS должно стоять перед любым знаком отношения;

блок_операторов-1 и блок_операторов-n — один или несколько операторов в одной или нескольких строках.

SHARED переменная[()] [AS тип] [,переменная[()] [AS тип]]... Предоставляет процедурам доступ к переменным уровня модуля. Здесь переменная — имя переменной уровня модуля или статической переменной. Имя переменной может быть длиной до 40 символов и должно начинаться с буквы. Допустимые символы: A-Z, 0-9 и точка (.). AS тип — описывает тип данных переменной (INTEGER, LONG, SINGLE, DOUBLE, STRING или тип данных, определенный пользователем).

SHELL [строка_команд$]. Приостанавливает выполнение программ BASIC для запуска команд DOS или командного файла. Здесь строка_команд$ — имя команды DOS или командного файла. Выполнение вашей программы возобновляется по завершении выполнения команды DOS или командного файла. Если вы опустили строку команд, SHELL вызывает COMMAND.COM и выводит на экран командное приглашение DOS. Для возобновления выполнения Вашей программы используйте команду EXIT.

SLEEP [секунды&]. Приостанавливает выполнение программы. Здесь секунды& — число секунд приостановки программы. Если секунды& равны 0 или опущены, программа будет приостановлена до тех пор, пока вы не нажмете клавишу или не произойдет событие.

SOUND частота, длительность. Генерирует звук через динамик вашего компьютера. Здесь частота — частота звука в Гц; значения находятся в пределах от 37 до 32767; длительность — число тактов системных часов, в течение которых длится звук; значения — в диапазоне от 0 до 65535. Каждую секунду происходит 18.2 такта.

STATIC переменная[()] [AS тип] [,переменная[()] [AS тип]]... Делает переменную локальной для функции или процедуры и сохраняет ее значение между вызовами. Здесь переменная — имя переменной уровня модуля или статической переменной. AS тип — описывает тип данных переменной (INTEGER, LONG, SINGLE, DOUBLE, STRING или тип данных, определенный пользователем).

STOP. Останавливает программу. Ключевое слово STOP также приостанавливает отслеживание событий в следующих операторах: COM, ON COM, KEY, ON KEY, PEN, ON PEN, PLAY, ON PLAY, STRIG, ON STRIG,TIMER, ON TIMER.

STRIG(n%) ON | OFF | STOP. Включает, выключает или приостанавливает отслеживание события для джойстика. Здесь n% — значение, указывающее триггер джойстика.

SUB. Определяет процедуру SUB. Синтаксис:

SUBимя[(параметры)] [STATIC]
  [блок_операторов]

END SUB. Здесь имя — имя процедуры SUB, длиной до 40 символов, без суффикса типа данных; параметры — одна или несколько переменных, указывающих параметры, которые передаются в процедуру SUB при ее вызове; STATIC — указывает, что значения локальных переменных процедуры SUB сохраняются между вызовами функции. При вызове процедуры SUB вы можете указать, что значение аргумента не должно изменяться процедурой. Для этого заключите аргумент в круглые скобки.

SWAP переменная1, переменная2. Обменивает значения двух переменных, где переменная1 и переменная2 — две переменные одного и того же типа данных.

**TIME$** = строк_выражение$. Устанавливает текущее системное время в Вашем компьютере, строк_выражение$ — время в одном из следующих форматов.

TIMER (n%) ON | OFF | STOP. Включает, выключает или приостанавливает отслеживание события для таймера. При включенном отслеживании события ON TIMER обращается к подпрограмме, как только прошло указанное в переменной (n%) число секунд.

TRON и **TROFF** включают и выключают слежение за операторами программы. При отладке в QBASIC отпадает необходимость в этих операторах.

TYPE. Определяет тип данных, содержащий один или несколько элементов. Синтаксис:

TYPE тип_пользов
  имя_элемента AS тип
  [имя_элемента AS тип]

END TYPE. Здесь тип_пользов — имя определяемого типа данных. Имя может быть длиной до 40 символов и должно начинаться с буквы; имя_элемента — элемент типа данных, определенного пользователем; тип — тип элемента (INTEGER, LONG, SINGLE, DOUBLE, STRING или тип данных, определенный пользователем). Используйте DIM, REDIM, COMMON, STATIC или SHARED для создания переменной определенного пользователем типа данных.

UNLOCK [#]номер_файла% [,{запись& | [начало&] TO конец&}]. Отменяет ограничения, наложенные последним оператором LOCK. Здесь номер_файла% — номер открытого файла;запись& для файлов прямого доступа — номер закрываемой записи по отношению к первой записи в файле. Для бинарных файлов -. Номер закрываемого байта относительно первого байта в файле; начало& и конец& — номера первой и последней записи или байта в диапазоне закрываемых или открываемых записей или байтов. Для последовательных файлов LOCK и UNLOCK влияют на весь файл.

VIEW [[SCREEN] (x1!,y1!)-(x2!,y2!) [,[цвет%] [,граница%]]]. Определяет размер и положение области просмотра, где графика может быть выведена на экран. SCREEN — указывает, что координаты задаются относительно экрана, а не области просмотра; (x1!,y1!)-(x2!,y2!) — координаты диагонали противоположных углов области просмотра; цвет% — атрибут цвета, устанавливающий заполняющий цвет области просмотра; граница% — атрибут цвета, устанавливающий цвет границы области просмотра. Если все аргументы опущены, область просмотра — весь экран. Допустимые атрибуты цвета зависят от Вашего графического адаптера и режима экрана, установленного последним оператором SCREEN.

VIEW PRINT [верх% TO низ%]. Устанавливает на экране границы области просмотра текста. Здесь верх% — номер верхней строки области просмотра текста; низ% — номер нижней строки области просмотра текста. Если Вы опустили аргументы верх% и низ%, то VIEW PRINT отводит весь экран под область просмотра текста. Диапазон для верх% и низ% зависит от режима экрана.

WAIT номер_порта%, AND-выражение% [,XOR-выражение%]. Приостанавливает выполнение программы до ввода двоичного кода из порта ввода. Здесь номер_порта% — номер порта ввода. AND-выражение% — целое выражение, которое WAIT объединяет со значением двоичного кода, используя оператор AND. Если результат не равен нулю, WAIT останавливает использование порта. XOR-выражение% может быть использова-

но для включения/выключения битов линии в двоичном коде перед использованием AND.

WHILE условие ... WEND. Выполняет серию операторов, пока указанное условие верно.

WIDTH [столбцы%] [,строки%] | {#номер_файла% | устройство$}, столбцы% | LPRINT columns%. Устанавливает ширину строк вывода устройства (например, принтера) или файла, а также изменяет число строк и столбцов, отображаемых на экране. Здесь столбцы% — нужная ширина в столбцах. Ширина экрана должна быть 40 или 80 столбцов; строки% — нужная высота экрана в строках. Допустимые значения — 25, 30, 43, 50 или 60, в зависимости от типа видеоадаптера и режима экрана; #номер_файла% — номер открытого файла или устройства; устройство$ — имя устройства: SCRN:, COM1:, COM2:, LPT1:, LPT2:, LPT3:.

WINDOW [[SCREEN] (x1!,y1!)-(x2!,y2!)]. Определяет логическое пространство для текущей графической области просмотра. Используйте оператор WINDOW для задания своей собственной системы координат в области просмотра. SCREEN — инвертирует обычное направление декартовых координат Y на экране так, что значения Y увеличиваются сверху вниз на экране; (x1!,y1!) — логические координаты, соответствующие координатам верхнего левого экрана области просмотра; (x2!,y2!) — логические координаты, соответствующие координатам нижнего правого экрана области просмотра. WINDOW без аргументов выключает логическую систему координат. Используйте оператор VIEW для изменения размера области просмотра.

WRITE [[#]номер_файла%,] выражение. Записывает данные на экран или в последовательный файл. Здесь номер_файла% — номер открытого последовательного файла. Если номер файла опущен, WRITE выводит данные на экран; выражение — одна или несколько переменных или выражений, разделенных запятыми, значения которых записываются на экран или в файл. WRITE вставляет запятые между объектами и заключает в кавычки строки в той последовательности, в которой они написаны. WRITE записывает значения в файл в той форме, которая может быть считана оператором INPUT.

XOR исключающее «или» в булевых операциях.

Код Сообщение

1 — NEXT без FOR
2 — Синтаксическая ошибка
3 — RETURN без GOSUB
4 — Нет данных
5 — Неверный вызов функции
6 — Переполнение
7 — Не хватает памяти
8 — Метка не определена
9 — Индекс вне режима
10 — Повторяющееся определение
11 — Деление на ноль
12 — Ошибка в режиме управления
13 — Ошибка ввода
14 — В строке нет места
16 — Слишком сложная строковая формула
17 — Невозможно продолжить
18 — Функция не определена
19 — Нет RESUME
20 — RESUME без ошибки
24 — Устройство в тайм-ауте
25 — Ошибка устройства
26 — FOR без NEXT
27 — Нет бумаги
29 — WHILE без WEND
30 — WEND без WHILE
33 — Повторяющаяся метка
35 — Подпрограмма не определена
37 — Ошибка счетчика аргументов
38 — Массив не определен
40 — Необходима переменная
50 — Переполнение FIELD
51 — Внутренняя ошибка
52 — Плохое имя файла/плохой номер
53 — Файл не найден
54 — Плохой режим файла
55 — Файл уже открыт
56 — Оператор FIELD активен
57 — Ошибка в/вв устройства
58 — Файл уже существует
59 — Неверная длина записи
61 — Диск заполнен
62 — Ошибка:введен конец файла
63 — Неверный номер записи
64 — Плохое имя файла
67 — Слишком много файлов
68 — Устройство недоступно
69 — Переполнение буфера коммуникации
70 — Нет разрешения
71 — Ошибка формата диска
72 — Ошибка диска
73 — Недоступная возможность
74 — Переименование через диски
75 — Ошибка доступа к пути/файлу
76 — Путь не найден

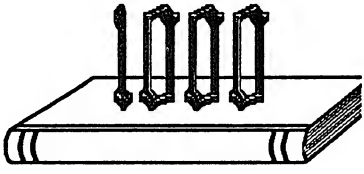
В этом списке нет позиций 0, 15, 21-23, 28, 31-32, 34, 36, 39, 41-49, 60, 65 и 66

Пришел **ЧЕРНЫЙ ДЕНЬ** для «ВТ и ее...» редакторов, получивших то-ли бессрочный отпуск, то-ли статус безработных. Читателям до января 1993 г. обещаны каникулы. Время такое, что планировать работу на два месяца вперед не позволяет себе даже фирма, «у которой нет проблем». Однако больше оптимизма — поступают различные предложения. Ведутся переговоры и есть надежда, что в розничной продаже появятся-таки наши брошюры. Издательская фирма, которая приютит коллектив редакции «ВТ и ее применение», еще не определена, но берусь утверждать, что оформление обложки сохранится (чего не скажу о названии), и наши выпуски легко распознает любой заинтересованный читатель.

СП Диалог

Акционерное общество
"ДиалогНаука"

Для изучающих иностранные языки предлагаем серию программ
Сергея Андреева



ТЫСЯЧА

ТЫСЯЧА предназначена для пополнения словарного запаса и проверки правильности написания слов. Рабочие языки программы: английский, французский, немецкий, испанский, итальянский и русский.

ТЫСЯЧА рассчитана на разные уровни знания языка: начальный, средний и уровень углубленного изучения отдельных тем. Разные уровни владения языком соответствуют разным наборам используемых слов. Начальный уровень - это около 1200 наиболее употребительных слов, которые для удобства пользования распределены по 40 различным темам (дома, в саду, в школе, в магазине, профессии, действия и т.д.). Средний уровень - это около 2200 слов. Этот уровень предлагает как больший набор слов по тем же темам, так и новые темы, не вошедшие в начальный уровень (полиция, суд, в научной лаборатории, в театре и др.). Уровень углубленного изучения отдельных тем ориентирован на подробное знакомство с лексикой соответствующих разделов.

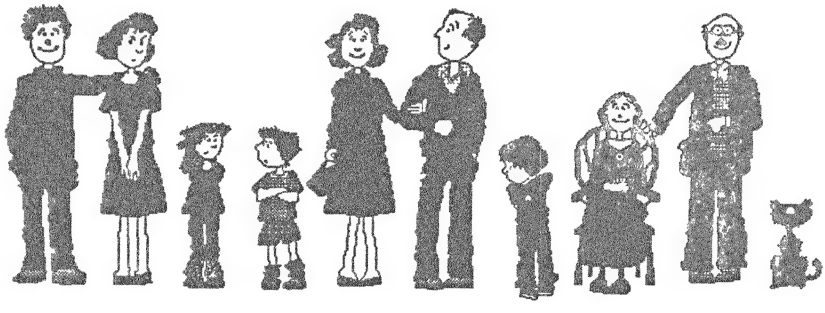
ТЫСЯЧА позволяет по-разному использовать имеющиеся в программе словари и рассчитана на разные режимы работы (в любом режиме возможно обращение к словарю за правильным переводом слова):

* выбор правильного перевода из набора предлагаемых вариантов;
* проверка правильности написания слов с различными вариантами подсказок (по первой и последней букве слова и числу промежуточных букв, только по первой букве слова);
* режим набора слов с клавиатуры.

ТЫСЯЧА поможет общаться с пользователями на любом из своих рабочих языков. Язык вопросов и язык ответов определяются независимо из этого же набора рабочих языков. Программа придет на помощь студентам, школьникам, а также иностранцам, изучающим русский язык.

ТЫСЯЧА крайне проста в работе, все управление осуществляется с помощью "мыши" или клавиатуры. Программу можно использовать

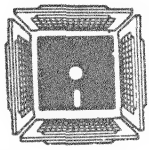
даже для обучения детей, которые еще не умеют писать, но умеют читать. Имеется свободно распространяемая демонстрационная версия.
ТЫСЯЧА предназначена для использования на IBM PC/XT/AT-совместимых компьютерах при наличии VGA или EGA адаптера. В составе поставки - одна ключевая дискета с программой, книга с документацией, специальная упаковка, лицензионный договор и регистрационная карточка пользователя.



фрагмент экрана при изучении темы "Семья"

Цена программы с учетом 28% налога: начальный уровень, включающий два языка, - 896 руб., добавление еще одного языка - 128 руб.; средний уровень (только англо-русский) - 1152 руб. Продажа школам - со значительной скидкой.

- Перечисленные выше программные продукты могут быть приобретены за наличный расчет (при этом предоставляется скидка 20%), по предоплате, по гарантийному письму или по договору.
- Для приобретения необходимого продукта по предоплате переведите требуемую сумму с указанием предмета платежа (например, "Оплата за программу **ТЫСЯЧА**") на наш счет.
- Перешлите нам по почте, факсу или передайте лично копию платежного поручения с исполнением банка.
- Банковские реквизиты: **получатель** - АО "ДиалогНаука", расчетный счет 467125; **банк получателя** - коммерческий банк "ДиалогБанк" в РКЦ ГУ ЦБР г.Москвы; **корреспондентский счет** 161625; **МФО** 201791; **код** 83.
- Почтовый адрес: 117967 Москва, ГСП-1, ул. Вавилова 40.
- Телефоны: (095) 137 0150 - директор, бухгалтер (и факс); 135 6253 - программные продукты и аппаратура; 137 0184 - антивирусные программы AIDSTEST и ADINF.
- Проезд: от станции метро "Ленинский проспект" любым трамваем или автобусом 115 до остановки "Улица Губкина" (4-я в направлении от центра города).
- BBS: (095) 135 6253 (с 20:00 до 10:00), FIDO 2:5020/6.13.
- Электронная почта RELCOM: THOUSAND@DIALS. MSK. SU.



Новая операционная система — это всякий раз очень важное и знаменательное событие, которого с нетерпением ожидают не только и не столько пользователи, сколько все, кто так или иначе причастен к компьютерной индустрии.

Александр Петроченков

РАЗРЕШИТЕ ПРЕДСТАВИТЬСЯ: DOS 5.0!

Долгожданная DOS 5.0, о выпуске которой пишет сегодня вся мировая компьютерная пресса, впервые появилась в продаже в июле этого года. Тогда же пришли первые сообщения, что фирмы IBM и Compaq приобрели право выпуска собственных модификаций DOS 5.0, а сотни фирм помельче уже получили право выпуска этой операционной системы по лицензии в составе своих компьютерных систем. Многие фирмы, выпускающие прикладное программное обеспечение, синхронно откликнулись выпуском своих новых продуктов, обеспечивающих работу в среде новой операционной системы. Одними из первых были фирмы Symantec с сервисным пакетом Norton Utilities 6.0 и Central Point Software с конкурирующим пакетом PC Tools 7.0, которая, кстати, передала Microsoft свою программу восстановления стертых файлов, полная версия которой входит в пакет PC Tools 7.0. В обмен на это Central Point Software получила право использовать в своем новом инструментальном пакете интерфейс, практически аналогичный оболочке MS-DOS Shell 5.0. Фирмы Novell и Banyan Systems сообщили, что их сетевое обеспечение также поддерживает DOS 5.0.

По сведениям, поступающим из московского представительства корпорации Microsoft, завершена работа над русским исполнением новой версии операционной системы MS-DOS 5.0. На европейском заводе Microsoft в Ирландии уже приступили к производству этого нового программного продукта и в ближайшие месяцы он поступит в продажу в нашей стране. Оперативность Microsoft в изготовлении русской версии может вселить надежду, что необузданный беспредел нашего дикого пиратского копирования программ пока не остановил сизифовы усилия этой могущественной корпорации по приобщению нас к самой передовой высокотехнологической цивилизации и обеспечению отечественных пользователей русскими версиями своих новейших программных продуктов. Однако долго ли продлится такая бескорыстная благотворительность? Кажется, вовсе не нужно быть Нострадамусом, чтобы с уверенностью предсказать, что вечной она не будет. А тем временем в зарубежную прессу уже проникают непроверенные сообщения, что Microsoft не остановилась на достигнутом и уже на всех парах разрабатывает следующую версию MS-DOS (шестую?), которая якобы будет принципиально отличаться от нынешней версии 5.0 и будто бы будет поддерживать новую файловую систему HEPS (High Performance Filt System операционной системы OS/2), а также интерфейсы DPMI (DOS Protected Mode Interace, реализованный в Windows 3.0 и в OS/2 2.0) и VCPI (Virtual Control Program Interface фирм Phar Lap и Guarterdeck), а кроме того, будет работать в защищенном режиме, в том числе и на процессорах Intel 8088.

Подтвердятся ли эти слухи и доведется ли нам когда-то увидеть следующую русскую версию MS-DOS 5.0, в процессе создания которой были учтены многочисленные справедливые замечания критиков, прозвучавшие в адрес предыдущей версии MS-DOS 4.01.

Что же нового в DOS 5.0? Кратко перечислить все небольшие изменения и улучшения едва ли возможно. В двух словах можно лишь сказать, что, хотя каких-то радикальных перемен не произошло, новая операционная система

стала гораздо удобней, более безопасной и комфортабельной для пользователей-новичков. Некоторые «профи» могут не без некоторого цинизма утверждать, что новая DOS утратила былое лидерство и плетется в хвосте тех технологических достижений, которые значительно раньше удалось реализовать другим разработчикам программных продуктов. Однако в оправдание Microsoft (разумеется, если деятельность Microsoft вообще нуждается в какой-либо апологии) могу заметить, что любая новая версия DOS должна быть чуточку консервативной, отбирая и интегрируя в себе лишь самые лучшие и нужные новшества и не бросаясь из стороны в сторону за яркими и блестящими безделушками. Впрочем, уже то, что новая операционная система, которой наверняка суждено серьезно изменить технологию управления миллионами персональных компьютеров, само по себе достаточно важная и злободневная новость. Поэтому лучше, наверное, все-таки начать с того, что в ней не изменилось, а соответствует нашим привычным представлениям об операционной системе DOS.

В этой версии MS-DOS в большей степени сохранена преемственность с предыдущими версиями. Сохранилась прежняя привычная таблица распределения файлов (FAT), позволяющая создавать файлы, имеющие восемь символов имени файла и три символа расширения имени. Имена каталогов также могут быть из восьми тех же самых символов ASII, которые использовались в прошлом. Командная строка по-прежнему может содержать максимум 126 символов. Уцелел без изменений и прежний архаичный командный язык пакетных BAT-файлов, о якобы предстоящем радикальном изменении которого распространялось немало слухов. Не оправдались и слухи, что в новой версии MS-DOS будет заменен файл COMMAND.COM по лицензии фирмы J.P. Software, которая широко известна своим популярным пакетом shareware-программ 4DOS, улучшающим и облегчающим работу пользователя в операционной системе DOS. Никаких заметных изменений не претерпела в MS-DOS 5.0 и поддержка последовательных периферийных устройств, ленточных накопителей, дисководов оптических дисков и принтеров. Впрочем, новинка все же есть: теперь поддерживаются дисководы для 3,5" флоппи-дисков емкостью 2,88 Мбайт. Однако жесткие диски, имеющие более 1024 цилиндров, по-прежнему никак не распознаются новой операционной системой.

Одним из самых заметных положительных изменений в MS-DOS 5.0 можно считать новый метод размещения в памяти. Ядро новой DOS, то есть внутренняя резидентная часть операционной системы, управляющая процессами работы остальной операционной системы и распределяющая для них физические ресурсы машины, заметно сокращено в размерах в сравнении с MS-DOS 4.01. От этого, впрочем, было бы мало проку, если бы не еще одно новшество, оказавшееся самым сенсационным: во всех современных машинах и во многих старых (за исключением тех машин на процессоре 8088, которые не имеют расширенной памяти) ядро DOS размещается в так называемой верхней памяти в промежутке между 640 Кбайт и 1 Мбайт, освобождая для использования прикладными программами рабочее пространство до 620 Кбайт. Именно это замечательное свойство MS-DOS 5.0 однозначно гарантирует новому программному продукту самое массовое признание и распространение во всем мире. Несомненно, дополнительное рабочее пространство памяти окажется очень кстати всем пользователям, которые применяют такие «жадные до памяти» программы, как приложения для графической среды Windows 3.0, Ventura Publisher, FoxPro и многие другие прикладные программные продукты.

Кроме этого «памятного» момента, есть и другая хорошая новость, касающаяся распределения ресурсов дополнительной памяти. В MS-DOS 5.0 включено средство EMM386.EXE, аналогичное известной утилите GEMM-386 фирмы Guarterdeck и позволяющее пользователям машин с процессорами i386 и i486 с помощью команд DH (DEVICEHIGH) и LH (LOADHIGH) убирать драйверы устройств и резидент-

ные программы из рабочей части оперативной памяти в «верхнюю» дополнительную EMS-память. Так что, если вы еще не успели обзавестись пакетом DESGviem-386, в который входит замечательная утилита GEMM-386, новая версия MS-D0S 5.0 будет вашим пропуском в прекрасный новый мир самых высоких технологий управления памятью Впрочем, похоже, что использование столь изощренных постмодернистских технологий пока по карману лишь очень немногим отечественным пользователям ПК.

Версия MS-DOS	Максимальный объем рабочей памяти	Максимальный объем памяти при работе ПК в сети
3.3	575 Кбайт	490 Кбайт
4.01	565 Кбайт	480 Кбайт
5.0	621 Кбайт	613 Кбайт

Эти цифры приводятся Microsoft применительно к ПК на процессорах 386. В зависимости от конкретной конфигурации машины и используемого программного обеспечения возможны некоторые отклонения от указанных значений.

С оболочкой DOS Shell, которой практически пользовались лишь очень немногие владельцы предыдущей MS-DOS 4.01, предпочитая более привычный Norton Commander,произошли весьма заметные и существенные перемены. Создатели новой оболочки явно постарались учесть и ликвидировать прежние недостатки, чтобы сделать работу в новой оболочке более комфортной и приятной. Новая DOS Shell может работать как в текстовом, так и в графическом режиме, а ее пользовательский интерфейс явно испытал облагораживающее влияние стиля Windows. Вначале может показаться, что новая оболочка функционально весьма похожа на многочисленные программные продукты других фирм, выполняющих привычный круг задач по управлению файлами: просмотр иерархической структуры дерева каталогов и содержимого текстовых файлов, копирование, удаление и перемещение файлов с помощью мышки. Однако дальше начинаются различия: управление программами в новой DOS Shell аналогично принципам, используемым в Windows и в Presentation Manager OS/2. Вы можете не продираться сквозь раскидистое дерево каталогов и файлов, а оперировать лишь ограниченной группой необходимых прикладных программ в списке из их названий. Несомненно, для большинства рядовых пользователей такое управление программами интуитивно более понятно и просто, чем освоение иерархической древовидной структуры хранения файлов на жестком и других дисках.

Появилась в новой оболочке и модная ныне возможность осуществления многозадачного режима работы: так называемый переключатель задач (Task Swapper — в англоязычной версии DOS 5.0) позволяет пользователю оболочки DOS 5.0 мгновенно переключаться из одной прикладной программы в другую, например, переходить из Lotus 1-2 в MS Word или dBASE. Очень вероятно, что здесь компьютерные «профи», наверняка, снобистики запротестуют: в MS-DOS 5.0 нет никакой подлинной многозадачности, и эти уловки рассчитаны только на наивных простаков. Действительно, в отличие, скажем, от DESGview переключение задач в оболочке MS-DOS 5.0 основывается не на переключении в оперативной памяти, а только лишь на диске. Переключатель задач лишь симулирует многозадачность, хотя никакого одновременного выполнения задач на самом деле вовсе не происходит. Поэтому с помощью такой псевдомногозадачности нельзя, скажем, одновременно выполняя две программы, «вырезать» кусок изображения из одной прикладной программы и «вклеить» в другую, как это возможно с помощью DESGview, Software Carousel или Windows. Однако можно предположить, что большинству пользователей DOS настоящая многозадачность на самом деле не очень-то и нужна, тогда как реализованная в MS-DOS 5.0 рудиментарная псевдомногозадачность наверняка окажется удобной, полезной и продуктивной для многих неискушенных конечных пользователей. А профессионалы наверняка смогут использовать для реализации подлинной многозадачности другие специальные программные средства.

Команды DOS также испытали достаточно заметные перемены. Появились некоторые новые команды, а также был пополнен синтаксис многих старых команд. Помимо того, что в оболочке DOS имеется удобная справочная система, теперь и любая внешняя или внутренняя команда DOS откликается на аргумент /? в своем синтаксисе, выводя экран с кратким описанием и пояснением применения. Многим, наверное, приходилось видеть приклеенные на мониторе бумажки, которые служат шпаргалками забывчивым пользователям и на которых нередко записывается синтаксис даже таких нехитрых команд, как FORMAT или PRINT. Теперь же для эффективной работы со всеми командами DOS практически совершенно не нужно листать справочное руководство: ответы на любые вопросы всегда под рукой. Разумеется, такое замечательное справочное новшество DOS можно только приветствовать, лишь удивляясь, почему же этого не было сделано раньше?

Команда FORMAT с аргументом /g теперь позволяет осуществлять быстрое форматирование дисков, которые ранее уже были отформатированы. Быстрое форматирование обнуляет таблицу FAT распределения файлов и каталогов, но данные при этом остаются не тронутыми. Это, во-первых, заметно экономит время при переформатировании дисков, которые находятся в хорошем состоянии, а во-вторых, если пространство на диске позволяет, таблица FAT и корневой каталог сохраняются где-нибудь на диске в специальном скрытом файле, чтобы с помощью команды UNFORMAT вы смогли бы вновь при необходимости восстановить ее утраченную структуру. Более полная и мощная утилита UNFORMAT входит в пакет PC Tools 7.0, а в новой версии DOS она появилась по лицензии, предоставленной Microsoft фирмой Central Point Software.

Другая новинка, с которой также уже должны быть знакомы пользователи последних версий пакетов PC Tools и Norton Utilities, это резидентная программа MIRROR, предназначенная для улучшения условий сохранности данных на дисках. В специальном защищенном файле MIRROR.FIL постоянно отслеживается и сохраняется текущая файловая структура диска. Этот файл всегда присутствует в корневом каталоге жесткого диска и переписывается при каждой перегрузке компьютера, а его предыдущая резервная копия сохраняется в файле MIRROR.BAK. Если в AUTOEXEC.BAT указана команда MIRROR C:, в файле MIRROR.FIL сохраняется копия FAT диска C:. Но можно сохранить в MIRROR.FIL также отраженные образы и других дисков, указав их поименно. Например, так:

MIRROR C D: E:

Программа MIRROR может работать резидентно, постоянно отслеживая уничтожение файлов. Для этого достаточно выполнить команду MIRROR с аргументом /t. Файл MIRROR.FIL может быть затем использован сопутствующей утилитой UNDELETE для восстановления уничтоженных файлов, разумеется, если их место на диске не было уже использовано для записи других файлов. В отличие от ранних версий утилит типа UnErase из Norton Utilities с помощью UNDELETE и MIRROR нет нужды вспоминать и подставлять стертую первую букву в восстанавливаемом имени файла, так как файловая структура полностью отражается в MIRROR.FIL. Точно так же с помощью упомянутой ранее утилиты UNFORMAT с диска читается скрытый файл MIRROR.FIL, если применялось быстрое форматирование по команде FORMAT с аргументом /g, и поэтому все случайно уничтоженные при быстром форматировании дискет данные могут быть затем успешно восстановлены. А если MIRROR запускается с аргументом /partn, по которому сохраняется структура разбиения жесткого диска, с помощью утилиты UNFORMAT можно выкарабкаться из гораздо более серьезной переделки.

Во многих популярных утилитах и оболочках, применяемых сегодня пользователями ПК, есть возможность просматривать содержание диска или каталога, подбирая файлы по определенным параметрам. Утилита DIR но-

вой версии DOS также может теперь воспроизводить на экране перечень файлов, которые включаются в список или исключаются из него по желательным параметрам. Она также может сортировать файлы по имени, расширению, размеру или дате. Между прочим, имена файлов теперь воспроизводятся DIR не заглавными, а прописными буквами, то есть точно так же, как, например, в Norton Commander.

Кроме того, многие утилиты и оболочки, например, тот же Norton Commander, позволяют восстанавливать «историю» введенных команд, а также выполнять простые макрокоманды. Теперь, наконец-то, и DOS имеет такую возможность. Маленькая резидентная сервисная программа DOSKEY, впервые появившаяся в DOS 5.0 и занимающая в памяти всего 4 Кбайта, позволяет вывести из буфера памяти на экран список ранее введенных команд. Достаточно нажать функциональную клавишу F7, и вы получите на экране перечень всех команд, выполненных после включения компьютера. Используя клавиши управления курсором, можно затем выбрать нужную команду, частично отредактировать ее и выполнить снова. С помощью этой утилиты можно также выполнять макрокоманды с различными изменениями параметров и по нажатию установленных клавиш. Макрокомандам можно присваивать любые удобные для запоминания названия, а также можно объединять их в цепочки, используя в качестве разделителя макрокоманд команду $T. Например, если вы хотите, чтобы по макрокоманде ДИСК форматировалась, а затем проверялась дискета в дисководе A:, можно написать такую макрокоманду:

```
DOSKEY ДИСК=FORMAT A: $T
CHKDSK A:
```

Разумеется, допустимо писать команды не только заглавными, но прописными буквами. Для более сложных макрокоманд в синтаксисе DOSKEY могут применяться пакетные параметры $1-$9, которые эквивалентны фиктивным параметрам %1-9% в командных пакетных файлах. Например, если в вашем компьютере два дисковода для гибких дисков, предыдущую макрокоманду можно сделать более удобной, если написать:

```
DOSKEY ДИКС=FORMAT $1 $T
CHKDSK $1
```

Если вам теперь понадобится отформатировать и проверить дискету в дисководе B:, достаточно написать в командной строке ДИСК B: и нажать клавишу Enter. А проявив еще немножко фантазии, можно создать гораздо более сложные и изощренные макрокоманды, которые смогут значительно облегчить жизнь за клавиатурой ПК.

Еще одна радостная новость: давным-давно морально устаревший строчный редактор текстов EDLIN наконец-то ушел на покой. Место этого древнего музейного экспоната теперь занимает новый и вполне современный полноэкранный текстовый редактор EDIT, который наверняка многим придется по вкусу. Причем EDIT.COM просто-напросто запускает текстовый редактор в файле GBASIC.EXE, который имеет непосредственное происхождение от известного GuickBASIC. Вот и еще новость: GBASIC заменил в DOS 5.0 ушедший в отставку старый GW-BASIC. Впрочем, присутствие GBASIC.EXE является скорее ловким рекламным трюком, отнюдь не заменяющим настоящий GuickBASIC: хотя он и имеет аналогичную гипертекстовую справочную систему, отладчик и такие же приемы редактирования, но компилирует только в памяти. Впрочем, превосходный текстовый редактор GBASIC, который, кстати, удобно управляется мышкой, сполна оправдывает введение в DOS 5.0 этой демонстрационной программы.

MS-DOS 5.0 поддерживает гораздо большее по объему прямое разбиение внешних накопителей: до 2 Гбайт вместо прежних 32 Мбайт. Причем теперь не требуется специально загружать утилиту SHARE.EXE, как это было необходимо в VS-DOS 4.01, чтобы преодолеть пресловутый 32-мегабайтный барьер. А пользователям особых разбиений диска, осуществляемых программами Disk Manager и SpeedStor, теперь предоставляется новая версия драйверов DMDRVR.BIN и SSTOR.SYS. С новой операционной системой могут использоваться прежние

сетевые драйверы, но имеется также и новый их набор. Новое, полностью переработанное руководство пользователя подробно объясняет приемы оптимальной настройки системы, а в текстовых файлах README содержится дополнительная информация по преодолению возможных проблем аппаратной и программной несовместимости. Новая версия DOS 5.0 предназначена для любых персональных компьютеров с процессорами от 8088 до i486, с объемом оперативной памяти не менее 512 Кбайт, при установке на жестком диске занимает 2,8 Мбайта.

Выпуск новой версии MS-DOS — это всегда очень сложная и трудная задача. В процессе выпуска новой версии корпорация Microsoft более года работала в тесном контакте с семью тысячами пользователей и дистрибуторов программных продуктов, чтобы «выловить всех блох» и в дальнейшем обеспечить всему миру быстрый безболезненный переход на новую операционную систему. Испытательные бета-версии MS-DOS 5.0 были известны многим отечественным специалистам уже с конца 1990 года. Похоже, что эти усилия не пропали даром. Все, кому часто приходится иметь дело с установкой операционных систем на новые машины, OEM (Original Equipment Manufacturer — изготовители комплексного оборудования, укомплектованного компонентами, изготовленными другими фирмами), многочисленные консультанты, эксперты и специалисты скоро смогут увидеть и испытать конечный результат этих трудов. Нет сомнения, что новая версия DOS немедленно превратится в фактический стандарт, с которым должны будут считаться все вовлеченные в компьютерную индустрию.

По сведениям, полученным из московского представительства Microsoft, в первую очередь доступ к новой операционной системе получат отечественные OEM, которым будут предоставляться лицензии на производство русской MS-DOS 5.0 для комплектации выпускаемых ими персональных компьютеров. Однако русская MS-DOS 5.0 будет также передаваться и отдельным пакетом для всех желающих, что обычно не практикуется в других странах. Цена пакета MS-DOS 5.0 ориентировочно будет равна 100 долларам или 180 немецким маркам. Наверняка такой пакет можно будет купить и за соответствующее количество рублей. А сегодня Microsoft уже уведомляет своих дистрибуторов о распродаже залежалых пакетов MS-DOS 4.01, которые теперь спешно распродаются всего лишь за 90-95 немецких марок. Microsoft также предлагает комплектный вариант распродажи MS-DOS 4.01 вместе с русской версией MS Works 2.0 за 210 немецких марок, тогда как рекомендованная базовая цена одного лишь пакета MS Works 2.0 была первоначально установлена в 295 немецких марок.

НАУЧНО-ВНЕДРЕНЧЕСКОЕ ПРЕДПРИЯТИЕ

* *АКМЕЛ* *

предлагает широкую гамму услуг в области:

— редакционно-издательских систем и технологических процессов по выпуску различных видов изданий на базе IBM PC и APPLE MACINTOSH;
— создание оригинал-макетов и фотонабора изданий (включая цветоделение);
— изготовление цветных буклетов и реклам в режиме «срочно» до 100 экз.;
— полиграфического исполнения и сбыта издательской продукции;
— бухгалтерского учета и экономики;

а также: организует и проводит обучение в этих областях;
оказывает информационно-консультационные и посреднические услуги.

☎ **8.253 3-76-84**
г. Пушкино, Московская обл.

☎ **477-29-81; 466-29-96;**
г. Москва

ОБМЕН ОПЫТОМ

В этой статье автор ссылается на: А.П.Гармашев. О текстах подпрограмм в кодах для БЕЙСИКа БК0010 (ВТ, № 8, 1990, с.12) и В.В.Авсеев, А.В.Авсеев. Использование возможностей БЕЙСИКа в программах на языке АССЕМБЛЕРа.

М.В.Михулин (г.Таллин)

ПРОГРАММА DEC

В журналах «Вычислительная техника и ее применение» N 8, 12.90 поднимался вопрос о подпрограммах в кодах для Бейсика БК 0010. Предлагаемая мной программа также имеет к этому отношение.

Известно, что соединить программу, написанную на Бейсике с подпрограммой в кодах можно при помощи операторов DATA и POKE и потом обращаться к ней через USR. Названные операторы содержат текст подпрограммы в виде чисел, представленных в одной из трех систем счисления: десятичной, восьмеричной или шестнадцатиричной. Программирующим в кодах на БК наиболее удобна и привычна восьмиричная система, поэтому, часто подпрограммы записываются в операторы DATA и POKE именно в этой форме. Однако, учитывая малый объем ОЗУ БК, на мой взгляд, лучше записывать подпрограммы в кодах в десятичной форме. Экономия памяти получается на отсутствии знака &O.

Разумеется, если имеется уже записанная на МЛ подпрограмма в кодах, то можно воспользоваться для ее упаковки в операторы DATA программами из журналов «В.Т.» N 8, 12.90. Однако, если подпрограмма находится в листинге, то ее перевод из восьмеричной системы в десятичную затруднителен из-за отсутствия в Бейсик 24.07.86 команды перевода.

Моя программа предназначена для автоматического перевода чисел из восьмеричной формы в десятичную и записи их в операторы DATA.

Программа написана в машинных кодах и занимает адреса &O1000 - &O1204. Загружается в память с МЛ командой BLOAD «DEC»,R. Обязательно R! Признаком начала работы программы является погасание курсора, восстановить который можно клавишей СТОП. После этого приступаем к набору программы на Бейсике и операторов DATA. Набор строк Бейсик-программы производится обычным способом. Строка с оператором DATA набирается следующим образом: номер строки и DATA, как обычно, а перед вводом чисел нужно нажать АР2+ любая клавиша. После этого числа набираются в той же самой восьмеричной форме, но без знака &O!!! Друг от друга числа отделяются запятыми, как и положено по формату оператора.

После набора всех чисел оператора нажимаем, как обычно, клавишу ВВОД. Программа автоматически переведет числа в десятичную форму, запишет их в память и произведет выход в Бейсик для набора других строк программы.

На экране ничего не изменится. Числа будут изображены так, как они были введены с клавиатуры, но после выполнения команды просмотра LIST все числа в операторах DATA будут изображаться в десятичной форме.

Признаком работы программы будет переход курсора после выполнения команды ВВОД не на следующую строку, а через одну. При просмотре программы промежутков между строк не будет. Набор следующего оператора выполняется в приведенной выше последовательности.

Как видно из описания программа использует вектор &O274, заменяя находящийся в нем адрес на свой собственный. В силу этого, при наборе Бейсик-программ нельзя задействовать программируемые ключи и элементы псевдографики, обработка которых ведется по вектору &O274. Вернуться к исходному состоянию Бейсика можно: при работающей программе клавишей КТ или последовательностью АР2, далее любая клавиша, КТ.

Преимущество хранения подпрограммы в кодах в десятичной форме очевидно. Например, программа COM.ASC из журнала «В.Т.» N 3.89 после замены на десятичную форму, стала занимать память на 200 байт меньше. Результат команды FRE() после компиляции программы редактора спрайтов из «В.Т.» № 1.91 стал 868 вместо 158. Сокращается время набора с клавиатуры.

Как отмечалось, программа может работать с операторами DATA и POKE, а также с LET, DEFUSR, частично с OUT.

При работе с программой требуется внимательность и аккуратность, так как программа не допускает редактирования. При наборе неверного числа или цифры нужно выйти из программы через клавишу ВВОД и набрать строку заново.

Программа «DEC» использует подпрограммы Бейсика, а именно: запись данных в память совместно с редактором Бейсика с адреса &O134012, перевод чисел из восьмеричной формы в десятичную с адреса &O140536, восстановление стека &O121626 (входит в редактор Бейсика &O121052). Ввод восьмеричных чисел с клавиатуры аналогичен подпрограмме монитора &O100472, но в несколько измененном виде.

Для хранения набранных с клавиатуры восьмиричных чисел программа использует буфер с адреса &O400.

Программа «DEC»

Адрес	Код	Ассемблер	Комментарии
1000	12737	MOV# 1012, @#274	Запись в ячейку 274_8 адреса начала программы
1002	1012		
1004	274		
1006	4737	JSR R7, @# 134012	Обращение к подпрограмме Бейсика
1010	134012		
1012	10146	MOV R1, - (R6)	Запись в стек последней занятой ячейки буфера ввода Бейсика
1014	12702	MOV # 400, R2	Начальный адрес буфера программы
1016	400		
1020	5005	CLR R5	Ввод восьмеричных чисел оператора DATA с клавиатуры
1022	104006	EMT 6	
1024	104016	EMT 16	
1026	120027	CMPB RO, #54	Если (,), то число заносится в буфер с адреса 400
1030	54		
1032	1416	BEQ 1070	
1034	120027	CMPB RO, #3	Возвращение к исходному состоянию Бейсика через клавишу <КТ>
1036	3		
1040	1454	BEQ 1172	
1042	120027	CMPB RO, #12	Переход к обработке восьмеричных чисел и занесению полученных десятичных в память
1044	12		
1046	1412	BEQ 1074	
1050	162700	SUB #60, RO	
1052	60		
1054	100761	BMI 1020	
1056	6305	ASL R5	
1060	6305	ASL R5	
1062	6305	ASL R5	
1064	60005	ADD RO, R5	
1066	755	BR 1022	
1070	10522	MOV R5, (R2)+	Занесение числа в буфер программы и переход к набору следующего
1072	752	BR 1020	
1074	10522	MOV R5, (R2)+	Запись в буфер последнего числа

1076	10200	MOV R2, RO	Запоминание последней занятой ячейки буфера программы
1100	12702	MOV # 400, R2	Адрес буфера программы
1102	400		
1104	12205	MOV (R2)+, R5	Извлечение восьмеричного числа из буфера для обработки
1106	10246	MOV R2, - (R6)	Запоминание следующей ячейки буфера
1110	4737	JSR R7, @# 140536	Обращение к подпрограмме перевода восьмеричных чисел в десятичные
1112	140536		
1114	12602	MOV (R6)+, R2	Восстановление ячейки буфера программы
1116	12603	MOV (R6)+, R3	Восстановление последней ячейки буфера ввода Бейсика
1120	5305	DEC R5	
1122	121127	CMPB @RI, #40	Извлечение преобразованного числа из области преобразования с адреса 3022 и запись его в буфер ввода Бейсика
1124	40		
1126	1401	BEQ 1132	
1130	111123	MOVB @R1, (R3)+	
1132	5201	INC R1	
1134	20105	CMP R1, R5	
1136	1371	BNE 1122	
1140	20002	CMP R0, R2	Проверка окончания буфера программы
1142	1404	BEQ 1154	
1144	112723	MOVB #54, (R3)+	Разделительная (,) между числами
1146	54		
1150	10346	MOV R3, - (R6)	Запоминание ячейки буфера ввода Бейсика и переход к обработке следующего числа
1152	754	BR 1104	
1154	12700	MOV #12, RO	Подготовка к выводу из программы и переходу к редактору Бейсика
1156	12		
1160	10301	MOV R3, R1	
1162	62706	ADD #16, R6	Возврат стека
1164	16		
1166	4737	JSR R7, @# 121626	Переход к редактору Бейсика
1170	121626		
1172	12737	MOV#101362, @# 274	Полный выход из программы и возвращение вектору 274 стандартного адреса 101362_8
1174	101362		
1176	274		
1200	62706	ADD #36, R6	Возврат стека
1202	36		
1204	700	BR 1006	Переход к редактору Бейсика

ДОСКА ОБЪЯВЛЕНИЙ

◇ *Интересуюсь возможностями МВЦ «Партнер»: работа с дисководом, принтером, цветное изображение, звук, обменяюсь программами.*
396070, Воронежская обл., Каширский р-н, ст.Колодезная, ул.Калинина, д.17 Гладырев В.И.

◇ *Имею ПО для ATARI 800 XL и SINKLAIR zx spectRum-48. Предлагаю обмен и деловое сотрудничество.*
446001, Самарская обл., г.Сызрань, ул.Ульяновская, д.119, кв.13 Позднякову Д.А.

◇ *Желаю приобрести программу позволяющую запускать программы для ZX-SPECTRUM (Синклер) на компьютере «Вектор 06Ц», а также другое программное обеспечение для данного компьютера.*
650069, г.Кемерово, пр.Химиков, 23а/252, Маркову Г., тел.51-16-73.

◇ *Хотел бы переписываться и обмениваться программами с владельцами БПЭВМ «Вектор-06Ц». Имею более 800 программ (200 в кодах), работающие квазидиск, дисковод, внешнее ПЗУ, принтер, джойстик.*
400066, г.Волгоград, ул.Краснознаменская, 8-63, Пономареву Александру Львовичу.

◇ *Хочу переписываться и обмениваться программами для ПЭВМ «Синклер Спектрум». Имею большой обменный фонд качество записи гарантирую.*
426039, г.Ижевск, ул.Дзержинского, д.17, кв.71, Винокурову Александру.

◇ *Всех, интересующихся компьютерной музыкой, приглашаем переписываться.*
183008, г.Мурманск, Молодежный проезд, д.6, кв.33, Пименов Геннадий Николаевич.
В портфеле:
— четырехголосный имитатор левой клавиатуры баяна (БК-аккомпаниатор);
— драйвер магнитофона на основе встроенного таймера (скорость до 8000 бит/сек).

◇ *Заочный Клуб Пользователей БК-0010 и БК-0010.01 «БЭКАШНИК» предлагает всем владельцам сотрудничество. Напишите нам и вы получите более подробную информацию.*
141008, Московская обл., г.Мытищи-8, а/я 94, Виноградову К.А., ЗКП «БЭКАШНИК».

◇ *Высылаю и обмениваюсь программами к компьютерам «АССИСТЕНТ» и «КОРВЕТ». Имею большой набор системных, прикладных, игровых, учебных программ. Желающим вышлю каталог.*
215010, г.Гагарин Смоленской области, ул.Гагарина, д.47 СПТУ 8 Орлову Б.А.

◇ *Предлагаю всем пользователям ПЭВМ «УКНЦ» ("Электроника МС 0511") новые оригинальные программы.*
346500, г.Шахты Ростовской области, до востребования Ключникову Вячеславу Юрьевичу.

Значительно повысить скорость, надежность и комфортность работы на БК можно, лишь подключив к нему дисковод. О том, как это сделать, и пойдет речь в этой статье.

А.М.Надежин

БК0010: РАБОТА С ДИСКОВОДОМ

Введение

В адрес БК0010 — поистине самого массового бытового компьютера в нашей стране было сказано немало хороших слов, в том числе и в журнале «Вычислительная техника и ее применение». За время выпуска БК, а это уже восемь лет, для него было написано множество программ разнообразного назначения. Многие из них по своим возможностям и даже дизайну не уступают программам профессиональных ЭВМ. Создано огромное количество текстовых и графических редакторов, систем управления базами данных и электронных таблиц, интерпретаторов и компиляторов различных языков программирования (от языка калькулятора МК61 до турбо-СИ), различных сервисных программ, облегчающих жизнь пользователю, ну и, конечно, компьютерных игр, количество которых уже перевалило за тысячу. Пользователями БК освоены различные периферийные устройства, такие, как джойстик, мышь, принтер.

Все приведенные выше аргументы доказывают, что БК0010 не бесполезная игрушка с ничтожно маленькой памятью, как утверждают некоторые (как правило, ни разу в жизни не работавшие на БК) критики, а достаточно хороший, мощный и удобный домашний персональный компьютер. Но разумеется, и БК не лишен недостатков. Основной из них, конечно, — малый объем памяти. Именно поэтому создание удобных и качественных программ для БК возможно только на языке Ассемблера, а для Ассемблера 16 КБайт не так уж и мало (если представить себе программу, занимающую всю память и состоящую только из откомпилированного ассемблерного текста, этот текст был бы размером около 8000 строк). Другой недостаток БК — использование бытового магнитофона в качестве внешнего запоминающего устройства. Конечно, для загрузки игровых программ магнитофон вполне приемлем, но при работе с текстами, а тем более при программировании на Ассемблере, возникают определенные неудобства. Так, у программистов, занимающихся реализацией на БК достаточно серьезных задач, на кассетах скапливается по сто и больше файлов с версиями исходного текста одной программы, не говоря уж о том, что сборка и перекомпиляция программы, состоящей, скажем, из пяти частей, записанных на магнитофон в разных файлах, весьма невеселое и долгое занятие.

Для облегчения участи пользователей, использующих БК не только как игрушку, были предприняты попытки создания электронного диска. В качестве примера можно привести расширение памяти и операционную систему RAMDOS, предложенные В.С.Коренковым (ИНФО N 6 за 1991 г. В.Коренков. Диспетчер ОЗУ и операционная система RAMDOS для БК-128К). Однако наличие электронного диска не избавляет пользователя от необходимости постоянно сбрасывать на магнитофон промежуточные результаты, ведь если этого не делать, может случиться так, что в один прекрасный момент у вас в доме отключат электричество, и все результаты трудов за день просто исчезнут. Значительно повысить скорость, надежность и комфортность работы на БК можно, лишь подключив к нему дисковод. О том, как это сделать, и пойдет речь в этой статье.

Итак, что же нужно, чтобы БК смогла работать с дисководом? Ну, разумеется, сама Бэкашка и дисковод. Но, я думаю, всем пользователям БК понятно, что напрямую дисковод к БК подключить нельзя. Для этого требуется специальное управляющее устройство — контроллер. Кроме того, потребуется операционная система — специальная программа, с помощью которой можно будет обращаться к дисководам, считывать и записывать файлы, переадресовывать обращения программ БК0010 с магнитофона на дисковод, выполнять сервисные функции: просмотр каталога, удаление, переименование, копирование файлов, форматирование дисков и т.д. Но как известно, многие программы БК0010 занимают всю память без остатка, поэтому потребуется еще и дополнительная память для хранения операционной системы и промежуточных данных (например, каталогов дисков).

Ну что ж, начнем по порядку. С БэКашкой все понятно, а какой же нам нужен дисковод?

Дисководы

Существуют три стандартных формата дисков: 3, 5 и 8 дюймов. Для наших целей больше всего подходят пятидюймовые дисководы, ведь восьмидюймовые весьма громоздки и давно уже не выпускаются, а трехдюймовые, наоборот, малогабаритны, удобны и перспективны, но, к сожалению, в нашей стране почти не выпускаются, а зарубежные модели пользователям БК совершенно недоступны (ввиду своей стоимости).

Существуют 40- и 80-дорожечные, одно- и двухсторонние пятидюймовые дисководы. Их можно использовать для подключения к БК, но более удобно использовать двухсторонние 80-дорожечные дисководы, так как они имеют наибольшую емкость (800 Кбайт) и распространены больше других.

Встречаются дисководы двух типоразмеров: «узкие» с передней панелью 150x42 мм (обычно 80-дорожечные, реже зарубежные 40-дорожечные), и «широкие» с передней панелью 150x85 мм (всегда 40-дорожечные). Из доступных дисководов наиболее надежны в работе зеленоградские: 80-дорожечные двухсторонние МС5305 и «широкие» двухсторонние 40-дорожечные МС5301. Несколько хуже армянские МС5305 и «широкие» немецкие (ГДР). Значительно хуже болгарские дисководы ISOT ЕС5321.

Итак, представим себе идеальную ситуацию: вы приобрели два роскошных зеленоградских МС5305, да еще и с блоком питания. Какой-же искать контроллер?

Контроллеры

Наиболее удобно воспользоваться готовым промышленным контроллером дисководов от ЭВМ с аналогичной архитектурой. Проще всего использовать контроллеры от БК0011 или БК0011М, однако подойдет и контроллер от УК-НЦ, но о нем чуть позже. Конструктивно контроллеры от БК0011 и БК0011М выполнены в стандартном корпусе блока МСТД и подключаются к разъему МПИ БК. Эти контроллеры построены на основе БИС 1801ВП1-128, поддерживают двойную плотность записи в стандарте MFM, аналогичную формату дисков IBM-совместимых компьютеров, и могут обслуживать один или два дисковода. Контроллер от БК0011 (блок КНГМД 3.059.001) отличается от контроллера БК0011М (блок КНГМД 3.059.014) только драйвером дисководов (программой управления), зашитым в ПЗУ (ПЗУ с драйверами для этих контроллеров называются соответственно 1801РЕ2-253 и 1801РЕ2-326). Более предпочтителен контроллер от БК0011М, так как его драйвер позволяет на БК0010 автоматически загружать с диска операционную систему. Для подключения этих контроллеров к БК требуется всего лишь установить перемычку между контактами разъема А19 и А2 и резистор сопротивлением 300 Ом между контактами А29 и А4 разъема МПИ контроллера. Резистор блокирует работу ПЗУ БК, расположенного по адресу 160000 (как раз к этим адресам и подключается ПЗУ контроллера).

Что касается контроллера от УК-НЦ (КМД УК), то для его подключения к БК потребуется больше времени, сил и средств: магистральные разъемы БК и УК-НЦ различны, поэтому необходимо подпаять контроллер к 64-контактному разъему БК; в контроллере от УК-НЦ нет тактового генератора, поэтому необходимо подать на него тактовую частоту 4 МГц от внешнего генератора; в контроллере от УК-НЦ нет ПЗУ с драйвером дисководов (но есть место под него), поэтому нужно установить в контроллер одну из двух вышеперечисленных БИС ПЗУ или загружать драйвер в дополнительное ОЗУ; некоторые разряды регистра управления у контроллеров от БК11 и УК-НЦ различны, поэтому нужно либо переделать контроллер от УК-НЦ в соответствии со схемой контроллера БК11, либо загрузить в ОЗУ специальный драйвер дисководов, переделанный под контроллер УК-НЦ.

Дополнительная память

Как уже говорилось, кроме дисковода и контроллера потребуется еще и дополнительное ОЗУ. Использование для этой цели ПЗУ невозможно, так как, кроме операционной системы, в дополнительной памяти должны размещаться переменные и буфер чтения/записи. В принципе возможен (и даже реализован в операционной системе MicroDOS) отказ от использования ОЗУ с помощью свопинга (сохранения на диске участков памяти с последующим их восстановлением), однако в этом случае значительно уменьшается скорость работы с диском, и все равно требуется использование нескольких ячеек основного ОЗУ для хранения переменных, и рабочей области контроллера дисководов. Из-за этого в упомянутой MicroDOS невозможна корректная работа программ с автозапуском и адресом загрузки ниже 1000. Правда, возможен еще один вариант — размещение операционной системы в ПЗУ, а переменных и буфера в дополнительном ОЗУ небольшого объема, но, на мой взгляд, проще и удобнее изготовить или приобрести блок дополнительного ОЗУ.

Дополнительное ОЗУ должно располагаться по адресу 120000 (при этом ПЗУ Бейсика или Фокала, естественно, должно быть отключено). Объем этого ОЗУ может быть 8 или 16 Кбайт. При объеме ДОЗУ 16 Кбайт первые 8 Кбайт (адреса 120000 — 137777) занимает операционная система, оставшиеся же 8 Кбайт (адреса 140000 — 157777) предоставлены пользователю. Наиболее предпочтительно использование статического ОЗУ с подхватом

питания. В этом случае достаточно будет один раз, загрузив с диска в ОЗУ операционную систему, пользоваться ей неограниченное время (время хранения информации в таком ОЗУ при выключенном питании ЭВМ может достигать нескольких месяцев).

Может возникнуть вопрос: не все ли равно, грузить систему с диска или хранить в ОЗУ. Дело в том, что непосредственная загрузка возможна лишь с помощью контроллера от БК0011М (ПЗУ 1801РЕ2-326), в этом случае из монитора БК дается команда старта по адресу 160000, а для того чтобы загрузиться с помощью контроллера от БК0011 (ПЗУ 1801РЕ2-253), потребуется сначала прочитать с магнитофона программу начального загрузчика и запустить ее. Эта необходимость вызвана тем, что в подпрограмме загрузки драйвера дисковода ПЗУ 1801РЕ2-253 используется системное прерывание ЕМТ-112, на БК0010 вызывающее обращение к адресу 10546 и зависание.

Чтобы избежать этого необходимо любым способом занести в указанную ячейку команду возврата из подпрограммы RET (код 207). Проще всего это сделать, прочитав с магнитофона программу начальной загрузки.

Существует несколько вариантов изготовления статического ОЗУ. Наиболее простой — ОЗУ на базе БИС 1809РУ1. В этом случае не потребуется никаких деталей, кроме самих микросхем ОЗУ и батарейки подхвата питания. Каждая такая БИС имеет емкость 2 Кбайт и подключается непосредственно к шине МПИ. Однако, несмотря на всю кажущуюся простоту, у этого варианта есть один существенный недостаток: БИС 1809РУ1 крайне дефицитны. Другой вариант — изготовление статического ОЗУ на базе логических микросхем распространенной серии 555 и микросхем ОЗУ серии 537. Несколько схем таких ОЗУ было опубликовано (ВТ N 11, 1991. Стр.40. Ю.Д.Хабибулин. Кассета ОЗУ ПЗУ для БК0010; ИНФО N 2, 1991. Стр.66. А.Барсуков. Энергонезависимое ОЗУ; ИНФО N 3, 1991. Стр. 51. А.Диков и др. TermOS). Существует и третий вариант — использование специализированного контроллера ОЗУ 588ВГ2 значительно менее дефицитного, чем 1809РУ1.

Схема такого ОЗУ, разработанная С.С.Авдюшиным, приведена на рис.1. Регистр адреса (DD1, DD2) выполнен на микросхемах 555ИР22 (можно использовать любые регистры с записью низким уровнем). Блок ОЗУ состоит из 16 микросхем 537РУ3 (537РУ2), которые могут быть заменены на 4 микросхемы 537РУ10 (537РУ8), включенные по схеме, показанной на рис.2. Формирователь на элементах DD3.1 DD3.2 предназначен для исключения ложных срабатываний регистров. Для повышения надежности работы контроллера дисководов его вход SYNC также можно подключить не непосредственно к разъему МПИ БК, а к выходу элемента DD3.2. Элементы DD3.3 DD3.4 служат для усиления мощности сигнала DOUT. На входах 12-14 контроллера DD4 задается адрес ОЗУ (при включении, показанном на схеме, адрес будет равен 120000). RC-цепочки на входах 2 и 17 контроллера нужны для формирования задержки сигнала RPLY на время задержки срабатывания микросхем памяти. Диоды VD1 и VD2 (обязательно германиевые) и элемент GB1 составляют схему подхвата питания микросхем ОЗУ. Кнопка S1 служит для перезапуска процессора в случае зависания программы или для выхода из программы в ОС. Кроме того, эта кнопка применяется для остановки процессора при включении и выключении ЭВМ (если во время подачи и снятия питания процессор не останавливать, может произойти разрушение информации в дополнительной памяти из-за выбросов управляющих сигналов). Резисторы R1 и R2 предназначены для отключения внутренних ПЗУ БК.

Ну что ж, теперь у нас есть БК, контроллер, дисковод, блок дополнительного ОЗУ. Это много, но еще не все. Нужна программа — операционная система, чтобы оживить все это нагромождение железа.

Операционная система ANDOS

Когда было принято решение о написании операционной системы, встал вопрос о формате записи на диск. Предполагалось использовать один из следующих форматов: формат RT11, формат MS-DOS или свой собственный формат с шестнадцатисимвольным именем. Но у всех трех вариантов были свои недостатки: Формат RT11 вводит серьезные ограничения на имя файла (имя не может быть больше шести символов плюс три символа расширения, причем символы должны быть допустимы в коде RADIX50, то есть заглавные латинские буквы и цифры), кроме того, в связи с последовательной записью файлов при удалении образуются пустые области, которые необходимо сжимать, а операция сжатия (по сути, переписывание заново всего диска) выполнялась бы на Бэ-Кашке с ее скудной памятью недопустимо долго. Новый формат записи, несмотря на все преимущества, имел серьезный недостаток: диски, записанные на БК в этом формате, не были бы совместимыми ни с какими другими ЭВМ (разработчики MicroDos выбрали как раз такой, свой формат).

Оставался последний вариант — формат записи MS-DOS, применяющийся в IBM-совместимых компьютерах. Единственным недостатком этого формата была сложность его реализации (фрагментирование файлов, обработка таблицы размещения), но с этим пришлось смириться. Зато формат MS-DOS открывал большие перспективы — возможность чтения и обработки на IBM дисков, записанных на БК, допустимость использования почти любых символов (в том числе и русских букв) в имени, возможность работать без сжатия диска.

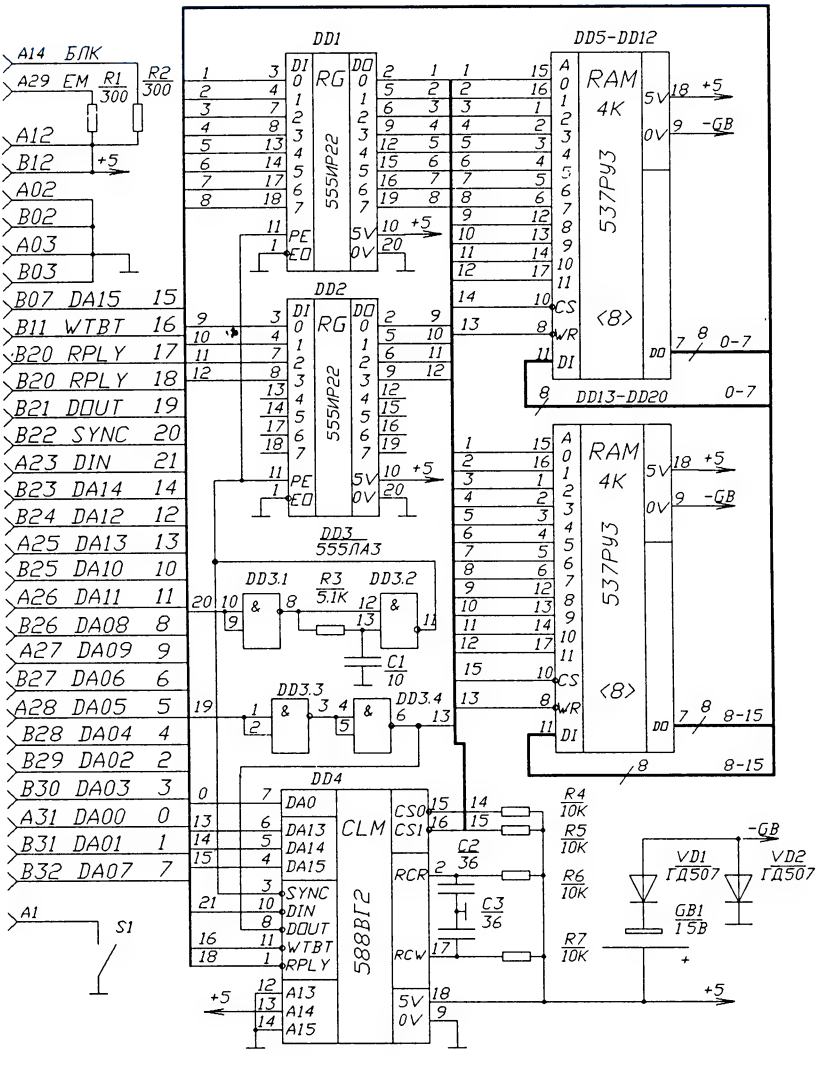


Рис. 1

Основа файловой системы была написана С.С.Авдюшиным в 1990 году, последующие исправления и дополнения внес автор этой статьи. Получившаяся операционная система получила название ANDOS. В апреле 1992 года была выпущена версия ANDOS V2.02$. О ней и пойдет речь. ANDOS V2.02$ может работать как на БК0010, так и на БК0011(М), причем в этом случае не потребуется дополнительного ОЗУ, и появится возможность работать с переключаемыми страницами памяти.

Итак, возможности ANDOS V2.02$:

- работа БК0010 как с магнитофоном, так и с двумя дисководами;
- работа БК0011 и БК0011М в режиме эмуляции БК0010;
- переадресовка обращений программ к магнитофону на дисковод;
- использование стандартного контроллера от БК0011 или БК0011М;
- использование стандартного формата дисков MS-DOS (файловая совместимость с IBM);
- возможность выполнения любых действий (просмотр каталога, удаление файла и т.д.), не выходя из программы пользователя;
- возможность сохранения страховочной копии (.BAK файла);

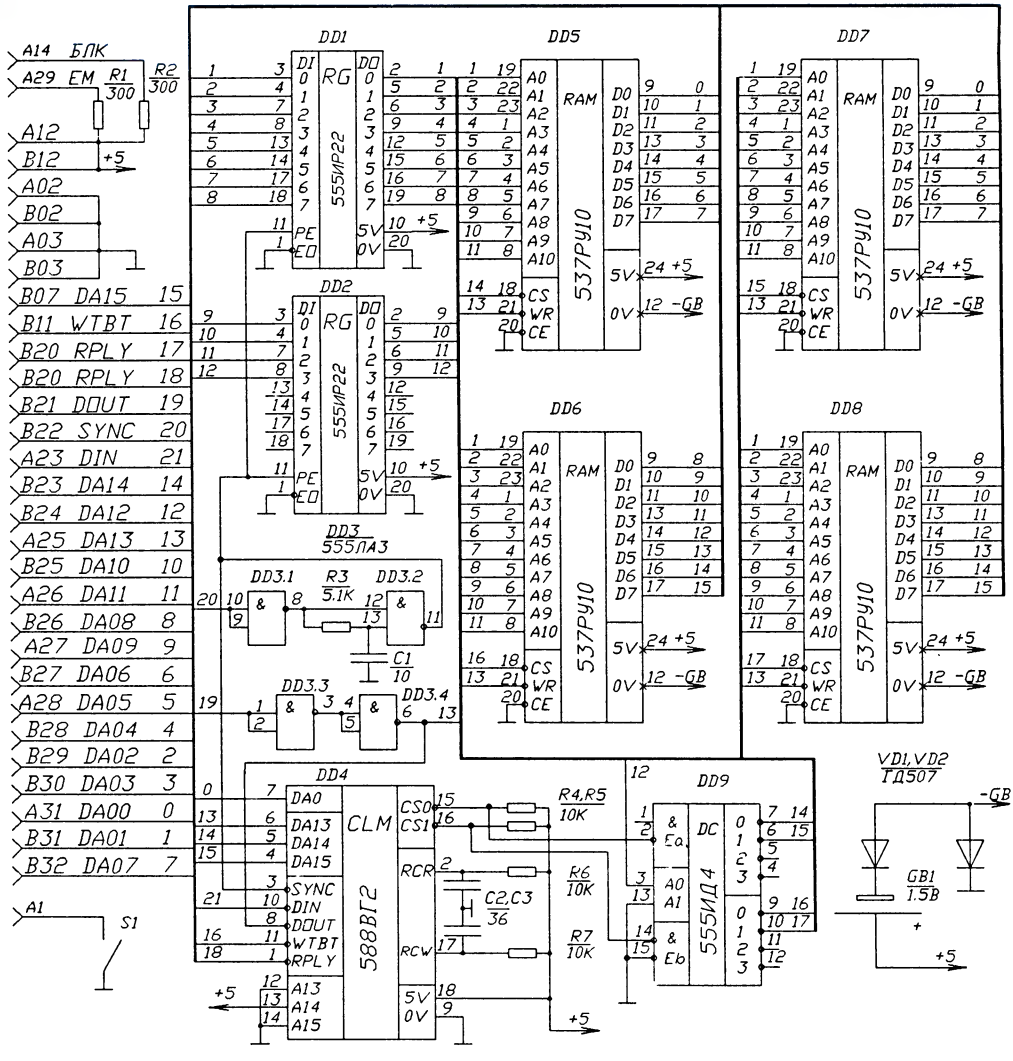


Рис. 2

- продуманная система защиты от случайных ошибок;
- копирование и запуск файлов длиной до 77460 байт;
- широкий набор системных утилит;
-высокая надежность обмена с диском;
- автоматическая настройка на любые типы дисководов;
- наличие логической обработки имени, позволяющей использовать имена вплоть до 16 символов и программы с защитой имени;
- корректная работа автозапуска в программах.

Формат дисков операционной системы ANDOS совместим с MS-DOS, но несколько отличается от формата «800». На диске может содержаться не более 112 файлов. Размер кластера (минимального объема информации, последовательно записываемого на диск) выбран равным 4 секторам (2 Кбайт). Это позволило сократить объем таблицы размещения файлов (FAT) до 1200 байт, повысить скорость чтения фрагментированных файлов и облегчить восстановление испорченных дисков.

Системный диск ANDOS снабжен самоадаптирующимся загрузчиком. При загрузке он определяет тип машины и выполняет соответствующие действия: при попытке загрузки на обычной БК0010 выдает сообщение о невозможности работы без дополнительной памяти, на БК0010 с допол-

нительным ОЗУ загружает операционную систему ANDOS по адресу 120000 и передает ей управление, на БК0011М устанавливает палитру БК0010, загружает монитор БК0010 по адресу 100000 и операционную систему по адресу 120000, после чего модифицирует в мониторе команду обращения к драйверу магнитофона, копирует монитор и операционную систему в четвертую страницу ОЗУ и запускает их. Загрузка на БК0011 отличается от загрузки на БК0011М тем, что стандартный монитор БК0010 модифицируется для обеспечения работы звука нажатия клавиши и записи на магнитофон.

После загрузки ANDOS выдает на экран заставку и ждет нажатия любой клавиши, после которого происходит инициализация системы и автоматическая адаптация под конфигурацию ЭВМ (если монитор находится в ПЗУ, происходит перехват вектора 30), и выход в монитор БК0010. Не удивляйтесь, ANDOS не имеет своего командного монитора и использует обычный монитор БК.

ANDOS V2.02$ обслуживает три устройства:

M: — Магнитофон

A: — Верхний дисковод (привод 0)

B: — Нижний дисковод (привод 1)

Для передачи всех команд ANDOS использует стандартный блок параметров магнитофона, поэтому все команды передаются через имя файла, так, например, для удаления файла перед именем нужно поставить знак «-».

Как и во всех операционных системах профессиональных ЭВМ в ANDOS имеется устройство по умолчанию. После загрузки это дисковод A:. Устройство по умолчанию можно изменить в любой момент вводом имени устройства вместо имени файла. При исполнении всех операций при вводе имени файла происходит обращение к устройству по умолчанию, если же перед именем файла явно указано устройство, происходит обращение к нему. Удаляется файл в случае, когда перед именем файла стоит знак минус. Если указано пустое (состоящее из одних пробелов) имя, на экран выводится каталог с указанием имен файлов, количества занимаемых кластеров, адресов и длин. В конце каталога выводится общее число файлов, количество занятых и свободных кластеров. Имеется возможность приостановить вывод каталога на экран, нажав любую клавишу, а также просматривать каталог файлов, имена которых начинаются на указанную букву алфавита.

При чтении файла с диска ANDOS ведет себя точно так же, как и монитор БК при чтении с магнитофона, поэтому в программах корректно работает автозапуск и проверка качества чтения.

Если при записи файла ANDOS обнаруживает на диске файл с таким же именем, на экран выводится запрос, в ответ на который можно отказаться от записи, переписать файл на место старого или перед записью заменить расширение у старого файла на .BAK и записать файл. Таким образом, в режиме дублирования появляется возможность всегда хранить на диске две копии файла — текущую и предыдущую (страховочную).

При ошибке обращения к диску выводится сообщение с кодом ошибки контроллера и предлагается повторить операцию или отказаться от нее.

При выполнении любой операции с диском ANDOS считывает содержимое загрузочного сектора (Boot Record), в котором записаны все параметры данного диска (общий объем, размеры каталога и FATа, количество сторон, объем кластера и другие параметры), и если эти параметры соответствуют формату записи ANDOS, настраивает под них драйвер дисковода, в противном случае выдается сообщение «Неправильный формат диска».

Для того чтобы, работая в ANDOS, можно было бы автоматизировать процесс переноса программ с кассеты на диск и использовать программы с защитой имени в ANDOS, применена логическая обработка имени: если имя превышает восемь символов, ANDOS использует первые семь символов и последний. Приведу пример. Допустим, вы захотели скопировать с магнитофона на диск игру «ДЕСАНТНИК». Она состоит из шести файлов: ДЕСАНТНИК, ДЕСАНТНИК1 ... ДЕСАНТНИК5. Если бы операционная система использовала просто первые восемь символов имени, все шесть файлов стали бы называться ДЕСАНТНИ и, естественно, затерли друг друга на диске. При копировании этих файлов в ANDOS с логической обработкой имени они получат имена ДЕСАНТНК, ДЕСАНТН1 ... ДЕСАНТН5. При запуске игры можно будет вводить как имя в каталоге (ДЕСАНТНК), так и полное имя (ДЕСАНТНИК). То же будет происходить и при чтении остальных пяти файлов в первой части игры (она даже и не узнает, что на самом-то деле их зовут совсем не так).

Дополнительное удобство ANDOS заключается в том, что для двух дисководов в памяти раздельно хранятся две таблицы параметров, что дает возможность подключать к БК два совершенно различных дисковода (например, один 80-дорожечный односторонний с быстрым приводом головки, а другой 40-дорожечный двухсторонний с медленным приводом).

Теперь хочется сказать несколько слов по поводу совместимости с IBM. Для успешного чтения на IBM диски ANDOS должны быть отформатированы на БК. (Диски, отформатированные на IBM, не читаются на БК по аппаратным причинам). К сожалению, у БК0010 и у IBM-совместимых компьютеров несколько разные требования к загрузчику, поэтому на IBM будут читаться только несистемные диски ANDOS. Чтобы прочитать системный диск, придется запустить специальную утилиту ANDOS, модифицировать загрузчик, а затем после работы на IBM восстановить его. Это связано с

тем, что для успешной загрузки на БК первой командой загрузчика должен быть NOP (код 240), а IBM при любом обращении к диску проверяет наличие в начале загрузчика команды перехода на подпрограмму загрузки JMP, и естественно, когда IBM обнаруживает в загрузчике БэКашный NOP, она говорит, что диск якобы не форматирован. Нормальное чтение дисков ANDOS на IBM возможно лишь на дисководе высокой плотности (1,2 Мбайт), при этом должен быть запущен какой-либо драйвер, позволяющий читать диски с плотностью записи 10 секторов на дорожку (800, PU_1700, FDREAD, DCE). Следует иметь в виду, что программа Norton Disk Doctor, обрабатывая диски ANDOS, будет выдавать сообщение о неправильном загрузчике (Invalid Boot Record), а в некоторых случаях еще и сообщение о неправильном типе диска (Media Discriptor). Это связано с тем, что диск у нас все-таки не IBMовский, а БэКашный, и ошибкой не будет. Следует иметь в виду, что ANDOS записывает адреса файлов в поле времени, поэтому при просмотре каталога ANDOSовского диска на IBM даты у всех файлов будут 01.01.1980, а времена произвольные (могут быть и несуществующие). Однако, как показали эксперименты, все программы IBM (в том числе и архиваторы) не изменяют времени создания файла, даже когда оно некорректно, следовательно, адреса БэКашных программ при копировании останутся неизменными. Для того чтобы на IBM можно было читать русские имена файлов и просматривать русские тексты, написанные на БК, был создан загрузчик шрифта КОИ8 в IBM. Правда, при его работе IBMовские программы, использующие псевдографику (например, Norton Commander), выглядят несколько странно, но с этим можно смириться.

Итак, для чего же может пригодиться совместимость с IBM? Ну, во-первых, для сохранения архива программ БК на дисках IBM (архиваторы, работающие на IBM, позволяют сжать даже самую плотную программу БК, как минимум в два раза). Во-вторых, для переноса текстов с БК на IBM и обратно (появляется возможность вводить тексты для IBM дома на БэКашке). В-третьих, для переноса графических картинок с IBM на БК (для этой цели А.С.Барским была написана программа, позволяющая записывать картинки, выведенные на экран IBM в экранном формате БК с перекодировкой цветов). В-четвертых, мощные программы IBM помогут обрабатывать диски ANDOS (С помощью Norton Unerase можно восстанавливать стертые файлы, уже упомянутым Norton Disk Doctor можно проверять и восстанавливать диски, программа Norton Disk Editor поможет внести исправления в загрузчик, каталог и FAT ну и, наконец, Norton Commander позволит вам наконец-то разобраться с архивом программ БК).

В комплект поставки ANDOS входит несколько системных утилит.

Утилита ANSETUP настраивает параметры ANDOS.

Утилита ANFORMAT позволяет форматировать, проверять и инициализировать дискеты.

Утилита CONBOOT позволяет преобразовывать загрузчик системного диска ANDOS в формат, читаемый IBM, и обратно.

Утилита ARC3 позволяет переписывать программы с кассеты на диск. Перезапись происходит в автоматическом режиме без ввода имен. Индицируется признак захвата файла, в случае обнаружения копии файла она пропускается. Имеется возможность просматривать заголовки файлов на кассете и копировать файлы длиной до 77460 байт.

Утилита DISCAS позволяет копировать одиночные файлы с диска на кассету или на другой диск.

Утилита ANPRIDIR позволяет распечатывать каталоги дисков на любых принтерах.

Утилита CONTXT конвертирует тексты из БэКашного формата в IBMовский и обратно.

Программа KOI8.COM загружает в IBM шрифт КОИ8.

Кроме того, разработана программа тиражирования кассет ANCAS. Она позволяет копировать с диска на кассету до 99 указанных файлов, причем возможно переименование каждого файла. Имеется возможность указывать скорость записи и число копий. Список файлов и переименований создается с помощью встроенного текстового редактора. Список файлов может быть сохранен на диске для последующего использования. Перед началом копирования можно выделить файлы из списка, не подлежащие копированию.

В заключение хочется сказать несколько слов о приобретении наших программных продуктов. Если вы решили приобрести операционную систему ANDOS V2.02$, программу ANCAS или драйвер дисководов, адаптированный под контроллер от УКНЦ, следует направить заявку по адресу: Москва 115573, А/Я 8, ANCO. В заявку вложите конверт с вашим адресом. Ответ с условиями поставки ANDOS V2.02$ вы получите не позже чем через две недели после получения заявки.

Хочется заранее предупредить всех будущих пользователей ANDOS: Зарегистрированные пользователи ANDOS получают право на письменные консультации и льготные условия приобретения новых версий ANDOS. Зарегистрированным считается пользователь, имеющий дискету ANDOS с фирменной наклейкой и серийным номером. Пользователь имеет право копирования операционной системы для своих целей, однако в случае распространения операционной системы он лишается права на авторское сопровождение.

Нам пишут

Ф.В.НОВИКОВ (г.Ярославль)

1

Часто бывает, что при отладке программ БК «зависает», и если программа не была записана, она безвозвратно теряется.

Чтобы избежать этого, надо установить переключатель, обеспечивающий перезапуск процессора. Его можно установить не разбирая компьютера, подсоединив контакты выключателя к контактам разъема МПИ А1 и А2.

Если теперь кратковременно замкнуть контакты и несколько раз быстро нажать стоп, то произойдет выход в монитор без потери программы. Выключатель удобно поместить вместо вставки на задней стороне блока МСТД. С помощью переключателя можно выходить из программ, блокирующих нажатие клавиши стоп, а также отсоединять блок МСТД без потери программы.

2

Программы, написанные на высоком уровне, часто начинаются с заставки. Сыграв несколько раз в одну и ту же игру, заставка приедается, однако приходится ждать ее окончания. Этого можно избежать, сделав так, чтобы по нажатию любой клавиши, программа переходила к игре. Но опрашивание клавиатуры после каждой операции займет слишком много времени и оперативной памяти.

Написав же маленькую подпрограмму, вы существенно увеличите удобство вашей игры.

```
MOV #ADK, @#60
.......
заставка
.......
ADR: MOV #IZN, @#60
.......
программа
.......
```

IZN — исходное значение, которое было в ячейке 60 до ее изменения;

ADR — адрес перехода после заставки.

Подпрограмма заносит по адресу вектора прерывания адрес, по которому должен произойти переход после нажатия клавиши. После перехода программа восстанавливает значение вектора прерывания от клавиатуры.

Д.С.СУХОВ (г.Сочи)

В одном из последних выпусков прочел статью с приведенной ниже программой о *HALP* драйвере. Но позволю себе заметить, что она немного великовата.

И я создал программу аналогичного характера, использующую под *HALP* пространство служебную строку. Замечу, что по желанию пользователя *HALP* информация может, естественно, выводиться в любой другой части экранной области и в «окне» любых размеров.

Привожу текст программы:

```
10 A$ (1...10)=STRING$ (30%,32%) +
    «текст пользователя»
100 FOR I%=1% TO (1...10)
110 FOR X%=1% TO LEN (A$ (I%))
120 POKE 130%, &O36100
130 ?AT(0%,0%) MID$ (A$ (I%), X, 32)
140 POKE 130%,&O40000
150 FOR T% = 0% TO (1%...+ ∞)
160 NEXT T%,X%,I%
```

Примечание: К 10-й строке — количество *HALP* информации может варьироваться и в более широких пределах, в случае нехватки памяти использовать оператор CLEAR. Подчеркнутое число изменяет величину «окна».

В.Б.ТАРАСОВ (г.Москва)

Предлагаю подключать джойстики к БК-0010.01 по следующей схеме:

«Земля» — А18 «Вправо» — В20
«Вверх» — В32 «Центр.огонь» — А24
«Вниз» — А-20 «Лев.огонь» — В24
«Влево» — А31 «Прав.огонь» — В29

По этой схеме джойстик работает почти со всеми программами, где он предусмотрен.

Э.В.НИЧИПУРЕНКО (г.Владивосток)

В журнале «ВТ» № 12 за 1990 г. были опубликованы две программы А.В.Алексеева которые позволяют упаковывать программы в машинных кодах в операторы DATA. Файлы, выдаваемые этими программами, можно сократить тем самым сэкономив память БК.

Я предлагаю упаковывать коды в операторах DATA плотнее за счет увеличения длины каждой строки с оператором DATA (максимальная длина строки 256 символов). Тем самым уменьшается количество номеров строк и слов «DATA».

Уплотнения также можно добиться за счет применения десятичных чисел без %. В данном случае числа с % занимают больше памяти, чем числа без %. Убедиться в этом можно на следующем примере:

```
10 DATA 1%,1%
20 ? FRE(1)
```

После запуска ЭВМ выдаст число оставшихся байт памяти равное 14024, если теперь строку 10 заменить на:

```
10 DATA 1,1
```

то после запуска ЭВМ выдаст число 14028. То есть экономим по два байта на каждом %.

Немного изменив алгоритм упаковки «ПРОГРАММЫ 1» на стр.37—38 я получил программу, которая работает по выше изложенному принципу. Текст программы прилагаю. Нумерацию строк я оставил прежнюю. Добавленные и измененные строки обведены.

```
10 CLEAR 7700
20 DIM A$(30)
30 POKE &O400,&O11505
40 POKE &O402, &O4737
50 POKE &O404, &O100536
60 POKE &O406, &O207
70 DEF USR=&O400
80 ? CHR$(140)+CHR$(140);
84 INPUT «НАЧАЛЬНЫЙ НОМЕР»; I%
86 INPUT «ШАГ СТРОК»; S%
88 LOCATE ,,1
90 A%=USR (&O42000)
100 C%=&O42000
110 DL%=C%+PEEK(&O266)-&O2
120 V%=1%
130 ? AT(10%,22%)"*"V%"*"
140  A$(V%)=""+MID$(STR$(I%),2%,LEN(STR$(I%))-1%)+"DATA"+CHR  $
(32%)
150 W$=""+MID$(STR$(PEEK(C%)),2%,LEN(STR$(PEEK(C%)))-2%)
160 IF LEN(A$(V%))+LEN(W$)>255% THEN 200 ELSE A$(V%) = ""+ A$ (V%)+
W$
170 IF LEN(A$(V%))=255% THEN 220 ELSE A$(V%)=""+A$(V%)+","
180 C%=C%+&O2
190 IF C%<DL% AND V%<31% THEN 150
200 A$(V%)=""+MID$(A$(V%),1%,LEN(A$(V%))-1%)
210 IF C%=DL% OR V%=30 THEN 250
220 V%=V%+1%
230 I%=I%+S%
240 GOTO 130
250 CLS
260 R$=""
270 INPUT «ИМЯ ФАЙЛА»; R$
280 ? «ВКЛЮЧИТЕ МАГНИТОФОН.»
290 IF INKEY$="" THEN 290
300 OPEN R$+".ASC" FOR OUTPUT
310 FOR J%=1 TO V%
320 PRINT # A$(J%)
330 NEXT J%
340 CLOSE
350 INPUT «БУДЕТЕ ПРОДОЛЖАТЬ»;U$
360 IF U$="D" OR U$="Д" THEN 80
370 CLS
380 END
```

Уважаемые коллеги!

Как мне представляется, для всех, кто в силу тех или иных причин проявляет интерес к современной компьютерной графике, заслуживающей внимание, будет информация о том, что в России готовится к изданию первый номер журнала **«КОМПЬЮТЕРНАЯ ГРАФИКА»**. Основные характеристики журнала:

- язык публикаций — русский
- объем — 4 печатных листа
- предполагаемая периодичность — 4 номера в год
- распространение — в России и других странах
- полиграфия — современная офсетная печать с воспроизведением в цвете
- предполагаемый выход в свет первого номера — июль 1992
- учредитель: Учебно-научный центр математического моделирования при Правлении общества «Знание» России, научный руководитель Центра — академик А.Самарский
- редколлегия: К.Гаунтлет (США) — заместитель главного редактора, В.Львов (Россия), Д.Арнольд (Англия), С.Клименко (Россия), Г.Нильсон (США), Т.Палташев (Россия), В.Крюгер (Германия), М.Грэйв (Франция), Б.Васильев (Россия), П.Вельтмандер (Россия), А.Кауфман (США), Н.Колтовой (Россия), Р.Хопгуд (Англия), Р.Сидорук (Россия), С.Чеботарев (Россия), Х.Хачен (Германия), О.Архипова (Россия), Е.Костогарова (Россия), Л.Трэйниш (США), Ф.Пост (Голландия), В.Ткачева (Россия) — ответственный секретарь.

Деятельность редакционной коллегии журнала осуществляется при поддержке и во взаимодействии с Российской академией наук, Национальной академией наук США, ACM/SIGGRAPH США, Российским международным фондом культуры, Еврографикой и рядом других крупнейших зарубежных фирм.

В журнале будут освещаться всевозможные аспекты современной мировой практики разработки и применения средств компьютерной графики в инженерной деятельности, научных исследованиях, менеджменте, искусстве, художественном дизайне и других областях. Должное внимание будет уделяться важному вопросу подготовки специалистов в области компьютерной графики. Специальный раздел журнала будет являться информационной базой для российских и зарубежных предпринимателей, проявляющих интерес к компьютерной графике. Предусмотрена специальная рубрика «Клуб читателей», предназначенная для оказания помощи в установлении коммуникаций среди российских и зарубежных читателей нашего журнала.

Стоимость первого номера журнала в России — 200 рублей, распространение — по предварительным заявкам. Для того чтобы заказать экземпляры первого номера, необходимо направить заявку по адресу: 101814, г.Москва, Новая площадь, 3/4, Правление общества «Знание», Архиповой Ольге Федоровне (контактный телефон — 925-36-34) и произвести денежное перечисление на р/с 161719 в Станкинбанк Главного управления Центрального банка России г.Москва МФО 201791 (для Учебно-научного центра математического моделирования (УНЦММ) — р/с 345 723 с указанием шифра журнала «КГ»).

В заявке необходимо указать наименование организации и ее представителя, на чье имя следует выслать журнал (или фамилию, имя и отчество частного лица, желающего приобрести журнал), индекс, полный почтовый адрес и контактный телефон.

Главный редактор
журнала «Компьютерная графика»
В.ПИЛЮГИН

Э 90 **Эти** разные Бейсики. — М.: Знание, 1992. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 4).
ISBN 5-07-002497-3

Рассказано от новой версии Бейсика (QBASIC) в структуре новой операционной системы — русифицированной MS DOS 5.0.

2404000000 **ББК 32.97**

Научно-популярное издание

ЭТИ РАЗНЫЕ БЕЙСИКИ

Редактор *Б.М.Васильев*
Мл. редактор *Н.А.Васильева*
Художник *В.Н.Конюхов*
Худож.редактор *И.А.Емельянова*
Техн.редактор *Т.В.Луговская*
Корректор *В.И.Гуляева*

ИБ № 12192

Подписано к печати 16.04.1992. Формат бумаги 70x100[1]/16. Бумага офсетная. Печать офсетная. Усл.печ.л.3,90. Усл.кр.-отт.8,45. Уч.-изд.л.5,52. Тираж 46511 экз. С—52. Заказ 96. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д.4. Индекс заказа 924704. Отпечатано с оригинал-макета издательства «Знание» на ордена Трудового Красного Знамени Тверском полиграфическом комбинате Министерства печати и информации Российской Федерации. 170024, г.Тверь, пр.Ленина,5.

Цена 90 коп.

Индекс 70195

Адрес подписчика:

Издательство «Знание»

Подписная научно-популярная серия

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ЕЕ ПРИМЕНЕНИЕ

Знаки коротко выражают и как бы отображают глубочайшую природу вещей, и при этом удивительным образом сокращается работа мышления.

Лейбниц

Цивилизация — это прежде всего информация, информация делает народы человечеством.

С.Залыгин

Наш адрес:
101835,
Москва,
Центр,
проезд
Серова,4